

Seagate Crystal Reports™ 6.0

Technical Reference

Seagate Software Information Management Group, Inc.
1095 West Pender St., 4th Floor
Vancouver, B.C., Canada V6E 2M6

© 1997 (manual and software) Seagate Software, Inc. All Rights Reserved.

Seagate Software, Seagate, and the Seagate logo are registered trademarks of Seagate Technology, Inc., or one of its subsidiaries. Seagate Crystal Reports, Seagate Crystal Info, the Seagate Crystal reports logo, and Smart Navigation are trademarks or registered trademarks of Seagate Software, Inc. All other product names referenced are believed to be the registered trademarks of their respective companies.

Manual written by
Gary Carter, John Morse, Lisa Barry, and Kirsten Sutton

INSTRUCTIONS ETC., INC.
134 W. Laurel Rd.
Bellingham, WA 98226
<http://www.instruct.com/>

1992-1997

C O N T E N T S

Welcome to the Seagate Crystal Reports Technical Reference

What's New	2
------------------	---

Chapter 1 - Crystal Web Report Server

Crystal Web Report Server Overview	10
Crystal Web Report Server Administration	10
Linking to Web Reports	12
Monitoring Web Activity with Reports	20
Sample Web Pages	23

Chapter 2 - Building Active Web Sites

Crystal Report Engine Automation Server	26
Visual InterDev Design-time ActiveX Control	26
Editing Active Server Pages	29
Sample Web Site	30

Chapter 3 - Configuring the Crystal Smart Viewers

Crystal Smart Viewer Overview	32
Crystal Smart Viewer/Java	33
Crystal Smart Viewer/ActiveX	34
Crystal Smart Viewer/HTML	39

Chapter 4 - Crystal Report Engine

Introduction to the Crystal Report Engine	44
Before using the Crystal Report Engine in your application	45
Using the Crystal Report Engine	46
Crystal Report Engine API	48
Exporting reports	68
Handling Preview Window Events	71
Distributing Crystal Report Engine Applications	76
Additional Sources of Information	76

Chapter 5 - Visual Basic Solutions

Using the Crystal Report Engine API in Visual Basic	78
Crystal ActiveX Control	82

Crystal Report Engine Automation Server	85
Working with Active Data	92
Crystal Data Object	100
Using Grid Controls with the Crystal Report Engine	102

Chapter 6 - Solutions for Other Development Tools

Using the Crystal Report Engine API in Delphi	108
The Crystal Visual Component Library	108
The Crystal Report Engine Class Library	110
The Crystal NewEra Class Library	112

Chapter 7 - Crystal Report Engine Reference

Working with section codes	116
Functions	121
Structures	353

Chapter 8 - Crystal ActiveX Control Reference

Section Codes (32-bit)	494
Section Codes (16-bit)	495
Properties	496
Methods	594

Chapter 9 - Crystal Report Engine Object Library

Object Hierarchy	610
Object Naming Conflicts	611
Object Library Events	611
Crystal Report Engine Object Library Reference	611
Error Codes	736
Report Engine Error Codes	737

Chapter 10 - Visual Component Library Reference

TCrpe Component	740
TCrpePrinterSetup Class	740
PROPERTIES	740
Methods	824
Events	836
Tutorials	837

Chapter 11 - Crystal Report Engine Class Library Reference	
Class Library Support Structures	931

Chapter 12 - Crystal Class Library for NewEra Reference

Chapter 13 - Creating User Defined Functions 1065

Designing User Defined Functions	1066
Programming the UFL	1069
Helper Modules	1069
Setting Up a UFL Project	1070
Function Definition	1071
Function Definition Table	1072
Function Templates Table	1073
Function Examples Table	1074
Error Table	1075
InitForJob Function	1076
TermForJob Function	1076
UFL Function Implementation	1077
Returning User Defined Errors	1077
Obtaining parameter values from the parameter block	1078
Picture Function - a sample UFL function	1079
Module Definition (.def) File	1082
UFJOB Modules	1083
Known problems with earlier versions of UFJob.C	1085

Chapter 14 - Creating User Defined Functions in Visual Basic

Creating User Defined Functions in Visual Basic	1088
---	------

Chapter 15 - Creating User Defined Functions in Delphi 3.0

Creating User Defined Functions in Delphi	1100
---	------

Index

Welcome to the Seagate Crystal Reports Technical Reference

What you will find in this chapter...

What's New, Page 2

Smart Runtime Preview Window, Page 2

Automation Server, Page 3

User-Defined Functions with COM, Page 3

Improved Support for Apex True DBGrid 5.0, Page 3

Reporting from Application Data, Page 3

Reporting Technology Designed for the Web, Page 4

Subreport Access from the ActiveX Control, Page 5

The Reporting Development Solution, Page 5

What's New

This version of Seagate Crystal Reports includes a large number of new features specifically designed for application developers and web administrators. Functionality has been added throughout to support many of the new user interface features in the program. In addition, developers working with ActiveX and the Component Object Model will be amazed at several new ActiveX components provided with this version of Seagate Crystal Reports.

If you build web sites, Seagate Crystal Reports is the reporting solution for your Internet or intranet site. The Crystal Web Report Server lets you start serving reports to your users in no time. For more powerful web site design, use the new developer tools available, including the Visual InterDev Design-time ActiveX Control, to build Active Server Pages for Microsoft IIS.

This section provides a brief description of all the features new to this release of Seagate Crystal Reports. All of the power and control over data access and reporting that you've come to expect in Seagate Crystal Reports leaps forward to provide the most advanced tools and techniques. Seagate Crystal Reports is your reporting solution.

Smart Runtime Preview Window

Seagate Crystal Reports now has a smart preview window that provides unprecedented interactivity. The result? Superior control over your application design, and a better experience with your application for your end users. Here are some of the new features of the smart preview window:

Smart Navigation

Give your end users a better way to interact with data. Smart Navigation offers powerful new tools to quickly navigate to the exact information you're looking for. Automatically generated group trees provide an index to your report so users can quickly jump to the section of the report they need to analyze with a single click of the mouse. In addition, you can now give users drill-down capability in the runtime preview window, the same functionality that's in the Crystal Report Designer.

Event and Call-back Support

New event support provides the ultimate interaction with your application through callbacks from the Crystal Report Engine. When a user clicks on an element in the preview window, your application can present additional report details, launch another report, execute application code, or do whatever you choose.

For complete information, see *Handling Preview Window Events, Page 71*, for the Crystal Report Engine API, and *Handling Preview Window Events, Page 89*, for the Crystal Report Engine Automation Server.

Additional Runtime Preview Window Functions

In addition to Smart Navigation and event/call-back support, the new runtime preview window includes a printer setup button, a variable zoom control, a refresh button, and search capabilities. Enable any or all of these new features, or continue using the current preview window - the choice is yours.

Automation Server

Seagate Crystal Reports contains an Automation Server that provides easy to use, object-oriented access to the entire Crystal Report Engine via over 500 properties, methods and events. If you're creating a project with Visual Basic or Office 97, this extensive support gives you the tools to tightly integrate and customize reports within your application.

For complete information, see *Crystal Report Engine Automation Server, Page 85*.

User-Defined Functions with COM

Do you need to add custom formulas to reports and find yourself limited to using C/C++? You won't any longer! Seagate Crystal Reports allows you to write your own formulas in any programming language that supports the Common Object Model (COM), including **Visual Basic**, **Visual J++**, **Visual C++**, and **Delphi 3.0**. Now it's fast and easy to perform customized calculations within your reports by writing your own UFLs.

For information on creating User-Defined Functions in Visual Basic, see *Creating User Defined Functions in Visual Basic, Page 1087*. For information on creating User-Defined Functions in Delphi 3.0, see *Creating User Defined Functions in Delphi 3.0, Page 1099*.

Improved Support for Apex True DBGrid 5.0

Do you want the reports within your application to accurately reflect the original format of the data within a grid? Now when you generate a report from Apex True DBGrid 5.0, grid formatting information is retained, including font types, styles and sizes. And you don't need to make any changes to your application! Just install the latest versions of the Crystal ActiveX Control and the Apex True DBGrid ActiveX Control, and your data will be printed better than ever before.

NOTE: *Seagate Crystal Reports does not include the Apex True DBGrid 5.0. You must be a licensed True DBGrid user.*

Reporting from Application Data

Need to connect to a data source on the fly? The new Active Data Driver in Seagate Crystal Reports allows you to create a report template that you can use within your VB application. Then, at runtime, use ADO, RDO, or DAO to bind your report to any runtime data source, including grid data. Or use the Crystal Data Object (CDO) to bind to application memory to report off memory blocks.

So now you need to deploy your reports to the Web. Time to pick up a whole new set of tools, right? Wrong! Seagate Crystal Reports contains new web reporting technology that's been designed to

provide you with the most advanced Web reporting capabilities available in the industry AND to allow you to transfer the reporting skills you already have to your Web projects.

Whether you are hosting your reports on an intranet or extranet, or integrating reports into a web-deployed application, Seagate Crystal Reports gives you the tools to get your Web project done right the first time, using skills and knowledge you already have.

For complete information on the Active Data Driver, see *Working with Active Data, Page 92*. For information on using CDO, see *Crystal Data Object, Page 100*.

Reporting Technology Designed for the Web

Want to deploy pages using your existing reports to your intranet but have a few concerns? Don't want to throw out all the reports that your organization is currently using? Concerned about having large reports shipped across a thin wire? Worried about your end users not being able to find the information they're looking for in an HTML report? Don't want the hassle of exporting all your reports to HTML just to be able to deploy them to your intranet?

Leverage Your Existing Reports

If you're using Seagate Crystal Reports, you already have many of the ingredients you need to deploy reports within your organization using the Web. Existing reports created with any version of Seagate Crystal Reports can continue to be used - just host them on your Web server, install and configure the Crystal Web Report Server (included in the box with Seagate Crystal Reports), and you're done! No web reporting tool available today gives you the power, convenience and flexibility of Seagate Crystal Reports.

Smart Navigation

For users, Smart Navigation makes finding the information they need fast and easy, whether it's on the desktop or within a browser. Rather than having to page through an entire report, users quickly navigate their way to specific details via a new page-on-demand interface that presents a summary of the report. You can also drill down for more detail on graphs, group totals, and embedded hyperlinks. Or use the convenient search tool to find what you're looking for.

Crystal Web Report Server

Even if you're deploying large reports to your intranet, the thin-wire architecture of the Crystal Web Report Server ensures superior response times and reduces web traffic: users can pull reports from the Crystal Web Report Server to the browser a page at a time as desired. An ISAPI/NSAPI-compliant component, the Crystal Web Report Server is compatible with all popular NT web servers including Microsoft Internet Information Server, Netscape Enterprise Server, and Lotus Domino. Regardless of your choice of Web server technology, Seagate Crystal Reports works seamlessly within your chosen environment to give you the best workgroup Web reporting technology available today.

Refer to *Crystal Web Report Server, Page 9*, for complete information on using the Crystal Web Report Server.

Crystal Smart Viewers

Distributing reports via the web doesn't mean you need to compromise when it comes to the presentation quality of your reports. With the Smart Viewer technology of Seagate Crystal Reports, you can deliver extraordinary reporting power and flexibility to users without the hassles of application setup on client desktops! If your browser supports ActiveX or Java, you can view reports in their *native* Seagate Crystal Reports format, meaning you can work with reports the same way on the web as you do on the desktop! In addition, Smart Viewers for HTML Frames or plain HTML gives users the ability to view *any* Crystal Report on *any* web browser. And all this without any intervention from you, the developer: you choose your web software, end users choose the browser they prefer, and Seagate Crystal Reports does the rest. This is the power and flexibility you can expect from the market leading reporting tool!

Configuring the Crystal Smart Viewers, Page 31, contains information about using the Crystal Smart Viewers.

Web Application Deployment

Do you want the same flexible reporting power in your web-deployed applications that you enjoy on the desktop? Seagate Crystal Reports includes the Crystal Report Engine Automation Server, allowing you to integrate the Crystal Report Engine directly into Active Server Pages. The Visual InterDev Design-time ActiveX Control makes the integration of reports into ASPs as simple as pointing and clicking!

See *Building Active Web Sites, Page 25*, for information on designing Active Server Pages using the Crystal Report Engine Automation Server and the Visual InterDev Design-time ActiveX Control.

Subreport Access from the ActiveX Control

If you're using the Crystal ActiveX Control to integrate reporting into your applications, you'll be pleased to know that you can now access subreport control through this interface, so the reports within your application can have even greater flexibility!

For a complete description of the ActiveX control, see *Crystal ActiveX Control, Page 82*.

The Reporting Development Solution

Whether it's web reporting that leverages your existing skills and knowledge, powerful and flexible reporting integrated tightly within your application, or presentation-quality reports from any database, you'll find that Seagate Crystal Reports is the one tool you need to get the job done fast, and done right.

The best creations are built with the best tools. With Seagate Crystal Reports, your toolbox just got bigger.

Part 1 - Crystal Web Solutions

1

Crystal Web Report Server

What you will find in this chapter...

Crystal Web Report Server Overview, Page 10

Crystal Web Report Server Administration, Page 10

Linking to Web Reports, Page 12

Monitoring Web Activity with Reports, Page 20

Sample Web Pages, Page 23

Crystal Web Report Server Overview



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

The Crystal Web Report Server is the reporting solution for Internet and intranet web sites running on Microsoft and Netscape web servers, or any web server that runs on the Microsoft Windows NT Server operating system. The Crystal Web Report Server provides the perfect interface for instantly displaying up-to-date reports in the familiar environment of the web browser. In addition, built-in reporting features allow enhanced web management through requests for information on activity logs generated by your web server software.

Crystal Web Report Server Administration

The Crystal Web Report Server can be installed on any Internet or intranet server running Microsoft Internet Information Server, Netscape FastTrack Server, or Netscape Enterprise Server. During installation of Seagate Crystal Reports,

- toggle the Web Server Only check box on in the Installation dialog box, or
- toggle the Local Install check box on in the Installation dialog box, then toggle the Custom installation check box on in the Installation Options dialog box, and finally, toggle the Crystal Web Report Server check box on in the Custom Installation Options dialog box.

The Seagate Crystal Reports setup application will automatically detect which web servers you have installed on the machine and will configure the web server appropriately.

Although the Crystal Web Report Server is installed with the most common settings selected, an interface is provided to allow you to make changes and customize your web server configuration. Through the Web Report Server Configuration application, you can edit the options for using and running the Crystal Web Report Server.

The Web Report Server Configuration application (WEBCONF.EXE) is installed, by default, in the \CRW directory. When run, the application displays a single dialog box with several tabs:



By making changes in this dialog box, you can customize the Crystal Web Report Server according to your needs. For complete documentation on the options available in this utility, simply run the application and press the F1 key at any time for context-sensitive Help.

NOTE: All settings made in the Web Report Server Configuration application are stored in the Windows NT Registry under the registry key HKEY_LOCAL_MACHINE\Software\Seagate Software\Crystal Reports\WebOptions. The descriptions that appear in the context-sensitive Help file include the appropriate registry value for each option. However, this is provided for informational purposes only. All values should be changed through the Web Report Server Configuration utility, not by directly editing the Registry.

Mime Types

Responses from the Crystal Web Report Server to web browsers contain a Mime type field. This field is used to determine the format of the message data. The following table shows the file formats with the corresponding Mime types:

<i>Mime Type</i>	<i>File Format</i>
rpt/x-epf	EPF File
rpt/x-ETF	ETF File
rpt/x-error	Error
text/html	HTML
text/plain	ASCII Text

Smart Navigation

Smart Navigation, a feature of the Crystal Smart Viewers, presents your users with a tree control much like the tree control in Windows Explorer. Smart Navigation, however, dynamically analyzes a report when it is first requested from the Crystal Web Report Server and populates the tree control with branches for each group in the report.

The Smart Navigation Group Tree in the Crystal Smart Viewers works like the Group Tree in the Seagate Crystal Reports Preview Tab. Simply expand and collapse branches in the Group Tree to find the section of the report you are most interested in. Click a branch to quickly jump to that part of the report.

Web administrators can control access to Smart Navigation and the Group Tree using the Group View options in the Web Report Server Configuration utility. The Configuration utility also allows you to control the maximum number of groups that are read into the Group Tree. If Smart Navigation is enabled, the Crystal Web Report Server must make an extra pass across the report to gather group information and generate the Group Tree. Such requests can tie up network resources and frustrate users, though, if large reports with several levels of grouping are being requested. Consider the information your users need, and decide if Smart Navigation is right for your web site.

Drilling Down on Data

A feature unique to the Crystal Web Report Server is the ability to perform drill down analysis on report data - to look at the details hidden behind subtotals and summary values. Users can click or double-click on summary values that allow drill down to display the detail values on a separate page inside a web browser or Crystal Smart Viewer. A simple summary report comprising only a few lines can be expanded to show all of the data used to derive the summaries.

As a web administrator, you can minimize hits on the database server by designing brief summary reports that allow selective drill-downs. Calculation of additional data is limited to specific user requests.

Linking to Web Reports

Pre-defined reports created with Seagate Crystal Reports are instantly available to any user connected to your web site via the Internet or an internal intranet.

- Internal reports are available throughout the company with point-and-click simplicity.
- Sales or Management staff can obtain up-to-date data through remote access while on the road.
- Corporate information can be published on an extranet for easy access by stockholders and potential investors.

As a web server administrator, you must determine how data is accessed from your web site and exactly how much of the data is available. The Crystal Web Report Server provides several commands that can be appended to URL requests in web page hyperlinks to help you control access

to reports and data. In addition, the Crystal Web Report Server provides the option of automatically prompting users for security information, stored procedure parameters, and parameter field values. Use this section as your toolbox for designing Crystal Web Report Server enabled web sites.

NOTE: The features described here allow you to control access to report information on a limited basis. Although the commands described in this section provide a certain level of customization, you should consider using the Crystal Report Engine Automation Server, Page 85, to design web sites if you need more control over report formatting and data at runtime.

Crystal Web Report Server Parameters

When requesting a report from the Crystal Web Report Server, or setting up a link to a report from another web page, there are several optional parameters available to customize the information returned. Parameters are passed with a report request by appending the URL address of the report with a question mark followed by each parameter you want to use. Parameters can be passed in any order and in any combination. All parameters are optional; if you do not specify any parameters, the original report will be returned.

The following is an example of using parameters when requesting a report:

```
http://www.domain.com/crweb/wsale.rpt?sf={customer.Sales}>10000&rf=1
```

Notice that each parameter is specified in the form:

```
parameter=value
```

«Where *parameter* is the name of the parameter, and *value* is the value you assign to that parameter. You should also notice that the parameter is preceded by a question mark (?) and multiple parameters are separated by an ampersand (&).»

The following parameters are available when requesting a report through the Crystal Web Report Server:

GF command

Specifies a group selection formula. This parameter is similar to *SF command, Page 16*, (selection formula). However, this parameter does not refresh report data if data is stored with the report.

```
GF=<formula>
```

«<>formula> is a selection formula in string format.»

For example:

```
GF= Sum( {customer.Sales}, {customer.Region})>10000
```

«Selects all groups where the sum of all customer sales in each region is greater than 10,000.»

For more information, see *Changing Selection Formulas in Web Reports, Page 17*.

INIT command

Specifies how the report should be displayed in the web browser. For example:

```
init=java
```

Possible values are:

- **java** - *Crystal Smart Viewer/Java, Page 33*
- **actx** - *Crystal Smart Viewer/ActiveX, Page 34*
- **html_frame** - *Crystal Smart Viewer/HTML, Page 39* (with frames)
- **html_page** - *Crystal Smart Viewer/HTML, Page 39* (standard)

If the INIT command is not specified, the Crystal Web Report Server will detect the type of browser requesting a report and will provide a default viewer for that browser. For instance, if the browser is Netscape Navigator 3.0, the Crystal Web Report Server will display the report using the Crystal Smart Viewer/Java.

NOTE: Not all browsers support all methods of viewing reports. For instance, versions of Netscape Navigator and Internet Explorer earlier than 3.0 do not support Java applets or ActiveX controls.

LOGDIR command

Specifies the directory of the web server log if the report is based on a web log file. This directory path can be a UNC path.

```
LOGDIR=<log_directory>
```

«<log_directory> is the full path to the log file directory in string format.»

For example:

```
logdir=c:\inetsrv\logdir  
logdir=\\SERVER\C\inetsrv\logdir
```

This command is only valid when you are requesting reports based on web server activity logs.

LOGS command

Specifies a date range for the web server log file to report on.

```
LOGS=<log_range>
```

«<log_range> is the date range of the log files to report on expressed in a string format.»

The following examples illustrate correct formats for the <log_range> string:

```
logs=19961231-19970115<STD ,ALL>
```

«Obtains IIS standard activity log information from December 31, 1996, to January 15, 1997.»

```
logs=19970101-19971231<NCSA,ALL>
```

«Obtains NCSA standard activity log information for the entire year of 1997.»

This command is only valid when you are requesting reports based on web server activity logs.

PASSWORD# command

Specifies passwords for logging on to SQL, ODBC, or password protected databases used by the report.

```
PASSWORD#=<password>
```

«<password> is a string.»

For example:

```
password0=secret
```

If the report accesses more than one password protected database, multiple passwords can be passed by incrementing the index number. For example:

```
password0=secret&password1=mystery&password2=unknown
```

PASSWORD# is normally used in conjunction with *USER# command*, *Page 16*. For example:

```
user0=SmithJ&password0=secret&user1=JohnS&password1=mystery
```

NOTE: *Make sure passwords appear in the URL in the same order that the password protected databases appear in the report. If passwords are not passed using the URL address, the user will be prompted for logon information at runtime.*

PROMPT# command

Specifies values for parameter fields in the report. Parameter field values are assigned to parameter fields in the order that the parameter fields exist in the report.

```
PROMPT#=<parameter_field_values>
```

«<parameter_field_values> is a string.»

For example:

```
prompt0=CA
```

If the report contains more than one parameter field, multiple parameter field values can be passed by incrementing the PROMPT index value. For example:

```
prompt0=CA&prompt1=1000
```

NOTE: *Make sure parameter field values appear in the URL in the same order that the parameter fields appear in the report. If parameter field values are not passed using the URL address, the user will be prompted to provide values at runtime.*

SF command

Specifies a selection formula. Will automatically refresh report data.

```
SF=<formula>
```

«<formula> is a selection formula in string format.»

For example:

```
http://server_name/reports/  
boxoffic.rpt?sf={studio.Studio}+%3d+"Universal"
```

«Selects all records where the studio is Universal.»

For more information, see *Changing Selection Formulas in Web Reports, Page 17*.

SPROC# command

Specifies values for stored procedure parameters accessed by the report. Values are assigned to stored procedure parameters in the order that the stored procedures are accessed by the report.

```
SPROC#=<stored_proc_values>
```

«<stored_proc_values> is a string.»

For example:

```
sproc0=CA
```

If the report accesses more than one stored procedure, multiple values can be passed by incrementing the SPROC index value. For example:

```
sproc0=CA&sproc1=1000
```

NOTE: Make sure stored procedure values appear in the URL in the same order that the stored procedures appear in the report. If stored procedure values are not passed using the URL address, the user will be prompted to provide values at runtime.

USER# command

Specifies user IDs for logging on to SQL, ODBC, or password protected databases used by the report.

```
USER#=<userids>
```

«<userids> is a string.»

For example:

```
user0=SmithJ
```

If the report accesses more than one password protected database, multiple user IDs can be passed by incrementing the USER index number. For example:

```
user0=SmithJ&user1=JohnS&user2=JSmith
```

USER# is normally used in conjunction with *PASSWORD# command, Page 15*. For example:

```
user0=SmithJ&password0=secret&user1=JohnS&password1=mystery
```

NOTE: Make sure user IDs appear in the URL in the same order that the password protected databases appear in the report. If user IDs are not passed using the URL address, the user will be prompted for logon information at runtime.

Refreshing Web Report Data

When a report contains saved data, the report does not need to access any databases when retrieved from the Crystal Web Report Server. This can greatly reduce network traffic and the use of network server resources when many people are frequently requesting reports. In fact, you may want to design most of your reports so that they contain saved data.

However, if a report contains saved data, and changes are made in the original database, the report will no longer reflect accurate information. To update the report, you can open it inside Seagate Crystal Reports, refresh the data, and save the report again. However, the Crystal Web Report Server also provides a means to dynamically refresh report data.

Changing Selection Formulas in Web Reports

In addition to specifying a record or group selection formula when designing a report, you can also change the selection formula using a command appended to the URL of a report called through the Crystal Web Report Server. As an administrator, you can create one report and design a web page that allows users to choose selection criteria for the information they need. The Crystal Web Report Server then dynamically generates the requested report with only the selected records.

To specify a record selection in a request for a web report, use the parameter *SF command, Page 16*. For example:

```
http://server_name/reports/boxoffic.rpt?  
sf={studio.Studio}+%3d+'Universal'
```

This will override any selection formula already contained in the report BOXOFFIC.RPT. However, the new selection formula will not be saved with the original report file. It is only valid for the currently requested job. The *GF command, Page 13*, can be used to change a group selection formula in a report.

The Crystal Web Report Server does not validate any selection formulas you send to a report. If the selection formula you create is invalid, an error will be returned to the web browser. If you are designing a web site that passes selection formulas to reports, be sure to test the selection formulas before allowing users to access your site.

SQL, ODBC, and Secured Databases

The Crystal Web Report Server opens reports based on SQL servers and ODBC data sources just as easily as reports based on smaller, desktop database files. If the data in a report requires access to a secure data source such as an SQL server or ODBC, the Crystal Web Report Server will automatically prompt any user to provide a user ID and password before it displays report data. The complex security that you carefully set up continues to work, even through the web.

NOTE: Although the Crystal Web Report Server requires users to log on before displaying reports that access secured databases, security conflicts can appear if several people attempt to access the same report simultaneously. To prevent such conflicts, you should add security to your web site that prevents users from having access to or seeing secured reports. Forcing users to log on to the Internet or intranet site is a common solution to providing complete system security.

The following image is an example of a log on page generated by the Crystal Web Report Server when encountering a report accessing secure data in a Microsoft SQL Server database. The log on page that appears to your users may appear slightly different depending on the type of data your reports are based on.

Please Provide the
Following Information

Database Logon

Server Name : SQL Server - Pub

Database Name : pubs

User ID : sa

Password :

OK

If you want to create links in your web pages that handle user IDs and passwords automatically, use the *USER# command, Page 16*, and *PASSWORD# command, Page 15*. These commands will let you specify more than one user ID and password if the report connects to two or more secured databases. Keep in mind that if an incorrect user ID or password is sent, the Crystal Web Report Server will report an error and prevent access to the report.

NOTE: The Crystal Web Report Server applies a simple encryption algorithm to user names and passwords. However, to ensure complete security of database access information you should make sure your Internet or intranet server has the Secure Sockets Layer (SSL) encryption protocol installed and enabled.

SQL Stored Procedures

If a report is based on SQL stored procedures, the Crystal Web Report Server can prompt the user for values to apply to stored procedure parameters. The Crystal Web Report Server recognizes the source of your data and provides all the interface you need for your Internet or intranet site.

The *SPROC# command*, *Page 16*, on the other hand, when placed at the end of a URL request for a report, can assign values to stored procedure parameters automatically. Use the SPROC# command when designing web sites that restrict users to only certain database values based on the stored procedure.

The Crystal Web Report Server does not validate any stored procedure parameter values you send. If the value you pass to the stored procedure is invalid, passing a string when a number is expected, for example, an error will be returned to the web browser. In addition, the Crystal Web Report Server does not allow you to change the format expected by stored procedure parameters. Be sure to test any web site that accesses reports with stored procedure parameters before allowing users to access that site.

Parameter Fields

Seagate Crystal Reports parameter fields are similar to SQL stored procedures but exist in the report file itself rather than the database. Once again, the Crystal Web Report Server can automatically prompt users for parameter field values using a dynamically generated web page similar to the following:



If you need to prevent users from specifying their own values for parameter fields, use the *PROMPT# command*, *Page 15*, with the URL specified in a hyperlink. PROMPT# lets you specify values for one or more parameter fields in a report.

The Crystal Web Report Server does not validate any parameter field values you send. If the value you pass to the parameter field is invalid, passing a string when a number is expected, for example, an error will be returned to the web browser. In addition, the Crystal Web Report Server does not allow you to change the format expected by parameter fields. Be sure to test any web site that accesses reports with parameter fields before allowing users to access that site.

Monitoring Web Activity with Reports

The Crystal Web Report Server not only allows end-user access to company data, but also works directly with your Microsoft or Netscape web server software to report on web activity logs. Webmasters can stay informed on important web site activity, even from a remote site. Using the Crystal Web Report Server you can quickly view and analyze web data through a connection to the web site itself.

If you manage a web site, the Crystal Web Report Server allows you to:

- instantly request reports on current web activity while on-line, and
- optionally limit information on web activity to a specific time period.

How to Design Web Activity Reports

Network administrators and webmasters often need important information about activity on the web server itself. Most common web servers provide this information in the form of a web activity log. Seagate Crystal Reports now includes a data source driver that allows you to design informative reports based on any NCSA-Standard Web Server Access Log.

Before you can design reports based on web server activity logs, you must make sure your web server is set up to produce activity logs, and you must know where those log files are being stored. For more information on setting up your web server to store activity log files, refer to the documentation for your web server software.

You design a web activity report just like you design any other report in Seagate Crystal Reports. The difference in a web activity report is that instead of gathering data from a database, you gather data from the web activity log generated by your web server.

- 1 Start Seagate Crystal Reports and click the New button on the standard toolbar. The Report Gallery appears.
- 2 Click a Report Expert in the Report Gallery. The Standard Expert will work for this example.
- 3 On the Data Tab of the Create Report Expert, scroll down and click on the data source button appropriate for your web server. For instance, if you are using Microsoft IIS as a web server, click the MS IIS Log button. If your web server produces NCSA standard web activity logs, click the Web Log button. The Select Log File Location dialog box appears.

NOTE: Your web server may store web activity information in SQL database tables instead of separate activity log files. If so, click the button appropriate to the location of your activity log information, and log on to the SQL database as you would log on to any standard SQL database.

- 4 Enter the path to the log files for your web server, and click OK. Use the Browse button, if necessary, to locate the correct directory.

NOTE: You may be required to specify more information for your web activity log, depending on the type of web server you specify.

- 5 Once you select a web log, the Choose SQL Table dialog box appears. The path to your web activity log files appears in the *SQL Databases* list box, and all available log files are listed in the *SQL Tables* list box. Highlight each log file you need to report on, clicking Add for each in turn. The log files will be added to the list box on the Data Tab of the Create Report Expert.

NOTE: Although your web activity log information may not be stored in a SQL database, access to activity log information is handled much the same way that access to SQL database data is handled in Seagate Crystal Reports. For this reason, the Choose SQL Table dialog box appears when you design a report based on web activity logs.

- 6 Click Done when you are finished selecting log files. If you select more than one log file, the Links Tab will appear in the Create Report Expert.
- 7 Click the Fields Tab in the Create Report Expert. The Database Fields list box displays all fields available in the selected web activity logs.
- 8 Select each field you want to appear on the report, and click Add. Fields added to the Report Fields list box will appear in the report.
- 9 Continue designing your web activity report just as you would design any other report using the Create Report Expert.
- 10 Click the Print Preview button on the standard toolbar when finished to view your report.
- 11 Click the Save button on the standard toolbar and save the new report file to a directory accessible from the web server.

For instance, you may want to store web activity reports in the \CRWEB directory under the web server root directory, or you may want to store them in a directory only accessible to web server administrators.

How to Request Web Activity Reports from your Browser

A web activity report can be requested from a web browser just like any other report is requested. Simply specify the path of the report file in the form of a URL address.

The Crystal Web Report Server also provides the LOGDIR and LOGS commands for use when requesting a web activity report. Use LOGDIR to change the directory to the location the web activity log files. Use LOGS to change the type of log file requested (STD or NCSA) and to specify a time period to report on. For more information on Crystal Web Report Server commands, see *LOGDIR command, Page 14*, *LOGS command, Page 14*, and *Crystal Web Report Server Parameters, Page 13*.

The Crystal Web Report Server includes two sample web activity report files:

- ACTIVITY.RPT
- WEBACCES.RPT

These reports are installed in the \CRW\REPORTS\101\INTERNET directory of your web server system. Before you can use the reports, however, you will need to change the location of the web activity log file that each report is based on. The reports were created in advance using a web server located at Seagate Software, IMG. You must change the reports to look for web activity logs on your own web server.

- 1 From Seagate Crystal Reports open one of the two sample web activity reports.
- 2 To use the reports on your own system, you must log on to the data source containing your own web activity log files. Choose the Log On Server command from the Database menu. The Log On Server dialog box appears.
- 3 Scroll through the Server Type list box until you find the data source appropriate to your web activity logs.
 - For example, if you store web activity information in an SQL database, such as Microsoft SQL Server or Oracle, select the correct database type for your database, such as ODBC - CRSS (Microsoft SQL Server via ODBC) or Oracle7 SQL (Oracle SQL version 7).
 - On the other hand, if you store web activity information in standard log files, either NCSA log files or IIS standard log files, scroll through the *Server Type* list box and select a data source appropriate to the web server format. For example, the MS IIS Log Files server type refers to Microsoft IIS activity log files. The Web Log Files server type, on the other hand, refers to NCSA standard web activity log files.
- 4 After you click OK in the Log On Server dialog box, a dialog box will appear that is appropriate to the type of data source you selected. If you selected an ODBC or SQL data source, for instance, the Log On SQL Server dialog box will appear and you can enter the information appropriate to logging on to your SQL server or ODBC data source.

Web server specific activity logs require information more specific to the web server. You will need to specify the location of the activity log files on disk, either on the local system or on your network. You may also need to specify the format of the log files and a time period during which log files were kept.

NOTE: For more information on how your web activity log files are stored, refer to your web server documentation.

- 5 When finished making changes, click OK and the program will log on to the web activity log data source. A message box appears indicating the log on attempt has succeeded. Simply click OK.

NOTE: If your log on attempt fails, repeat the log on procedure again, paying close attention to the information you enter in each dialog box. If your log on attempt continues to fail, contact your network administrator to verify you have proper access to the data source.

- 6 From the Database menu, choose the Set Location command. The Set Location dialog box appears.
- 7 Notice that under the Location section of the dialog box, the Server Name specifies the location of the activity log files when the report was originally created. Most likely, this server does not exist on your own network. Click the Set Location button to change this location. Since the report was originally set up on a Microsoft IIS server, the Select IIS Log Files and Dates dialog box appears.

- 8 The next step depends on the type of web server you have:
 - If your web server is a Microsoft IIS server and activity logs are stored as standard IIS Log files, use the Select IIS Log Files and Dates dialog box to change the location of the web activity logs. Click OK when finished.
 - If your web server is a Netscape server, or if you have set up your server to store activity logs in an SQL or ODBC database, click Cancel, and the Choose SQL Table dialog box will appear. Select the data source you logged on to earlier, and click OK.
- 9 The Server Name in the Set Location dialog box should now reflect the correct location of your web activity logs. Click Done.

After the Set Location dialog box closes, the program queries your web activity log files and updates the sample report. Save the sample report for future reference.

Sample Web Pages

When you install the Crystal Web Report Server on your Internet or intranet server system, several sample web pages are also installed. These pages allow you to browse through several sample reports using any of the Crystal Smart Viewers. Once you have installed the Crystal Web Report Server, simply open your web browser and enter the following URL:

`http://<server>/crweb/default.htm`

«<server> is the name of your web server domain.»

For example:

`www.img.seagatesoftware.com`

2

Building Active Web Sites

What you will find in this chapter...

Crystal Report Engine Automation Server, Page 26

Visual InterDev Design-time ActiveX Control, Page 26

Editing Active Server Pages, Page 29

Sample Web Site, Page 30

This chapter contains information specific to the Professional Edition of Seagate Crystal Reports.

Crystal Report Engine Automation Server

The Crystal Report Engine Automation Server allows you to build powerful active web sites with any web server that supports ActiveX and Active Server Pages. This section demonstrates how to bring the full power of the Crystal Report Engine Automation Server to your Microsoft Internet Information Server (IIS) 3.0 web sites.

Using the Crystal Report Engine Automation Server, you can design web sites that display powerful reports without using the *Crystal Web Report Server, Page 9*. Although the Crystal Web Report Server provides many features for viewing reports from a web browser, the ability to format and restrict data is limited to a few simple commands. The Crystal Report Engine Automation Server, on the other hand, gives you all of the power of the Crystal Report Engine.

To add the Crystal Report Engine Automation Server to a web site, Seagate Crystal Reports provides the Visual InterDev Design-time ActiveX Control for use with Microsoft Visual InterDev. In this chapter, you will learn how to use the Visual InterDev Design-time ActiveX Control to automatically set up VBScript code in your Active Server Pages that uses the Crystal Report Engine Automation Server and the Active Data Driver.

*NOTE: For complete information on the objects, methods, and properties provided by the Crystal Report Engine Automation Server, see *Crystal Report Engine Object Library, Page 609*.*

Visual InterDev Design-time ActiveX Control

When you install Seagate Crystal Reports, if you select the Custom installation option and choose to install the Development tools, the Crystal Report Engine Automation Server and the Visual InterDev Design-time ActiveX Control will be installed and registered on your system. To successfully use these tools with a web site, though, you must also install them on your Microsoft IIS web server system.

When used from within Active Server Pages, the Crystal Report Engine Automation Server dynamically produces reports that are displayed in a web browser using one of the Crystal Smart Viewers (see *Configuring the Crystal Smart Viewers, Page 31*). Using the powerful server-side scripting capabilities of Active Server Pages, you can design useful web-based interfaces and query tools for accessing reports over the Internet, an intranet, or an extranet.

The Visual InterDev Design-time ActiveX Control has been designed specifically for Microsoft Visual InterDev. Using the Visual InterDev Design-time ActiveX Control, you can quickly and easily add server-side VBScript code to your Active Server Pages. The design-time control lets you select an existing report, or build a dynamic report at runtime that accesses data from an ActiveX data source such as ActiveX Data Objects (ADO).

Selecting an Existing Report

This section demonstrates how to use the Visual InterDev Design-time ActiveX Control to add code to your Active Server Pages that displays an existing report in the Crystal Smart Viewer/Java.

- 1 With Microsoft Visual InterDev running and a web project open, choose the New command on the File menu. The New dialog box appears.
- 2 On the Files Tab of the New dialog box, select Active Server Page.
- 3 Enter a name for the new Active Server Page in the File name text box, and click OK. The new Active Server Page is created and appears in the Microsoft Developer Studio.
- 4 Make sure the comment <!-- Insert HTML here --> is highlighted, and choose the ActiveX Control command from the Insert | Into HTML menu. The Insert ActiveX Control dialog box appears.
- 5 Click the Design-Time Tab to display design-time ActiveX controls that are registered on your system.
- 6 Highlight the CrystalReport.DTC design-time control, and click OK.
- 7 Close the Properties dialog box if it appears. The Visual InterDev Design-time ActiveX Control appears in the Microsoft Developer Studio window.
- 8 Toggle the Use Existing Report check box on if it is not on already. This disables most of the controls in the design-time control. If you have not added a data connection to your web project, this check box will automatically be toggled on and disabled.
- 9 Click Browse to select the report you want to display in your web site. The Select Report Name dialog box appears.
- 10 Use the Select Report Name dialog box to locate and select the report you want to use. This should be a standard DOS path to the report. Do not use a URL address or a UNC path. Click Open when finished. The selected report appears in the Report Name text box of the design-time control.
- 11 In the Viewer drop-down box, select the Crystal Smart Viewer you want to be used to display the report.
- 12 When finished, close the window that contains the Visual InterDev Design-time ActiveX Control.

Once you close the design-time control, the ReportServer Active Server Page (RPTSERVER.ASP) is added to your project, along with the report file you selected in the design-time control. The ReportServer Active Server Page contains the VBScript code that communicates with the Crystal Report Engine Automation Server. This page is required by any web project that uses the Crystal Report Engine Automation Server to display reports.

In addition, the design-time control adds several lines of VBScript code to your web page. This code is designed to work with the ReportServer Active Server Page and the Crystal Report Engine Automation Server to display the report you selected inside the Crystal Smart Viewer you selected. For information on how to edit this code, see *Editing Active Server Pages*, Page 29.

To test your new Active Server Page, save the file to your web server, select the page in your Project Workspace window, and choose the Preview in Browser command from the File menu. Visual InterDev will open your web browser and display the Active Server Page containing the instructions to open the report.

Building a Report at Runtime

The Visual InterDev Design-time ActiveX Control can build a report dynamically at runtime based on an ActiveX data source such as ActiveX Data Objects. The design-time control uses the Crystal Active Data Driver to dynamically design a generic report based on tables and fields you select from a data source connected to your project. To use the dynamic reporting features of the design-time control, you must start by adding a data source to your web project.

- 1 With a project open in Visual InterDev, choose the Data Connection command from the Project | Add To Project menu. The Select Data Source dialog box appears.
- 2 To select an ODBC data source, click the Machine Data Source Tab.
- 3 Select a data source from the Data Source Name list, and click OK, or click New to create a new data source. Once you click OK in the Select Data Source dialog box, the data source will be added to your project. Save your project.

NOTE: You may be required to specify log on information when you select a data source. For complete information on adding data sources to a web project, refer to your Visual InterDev documentation.

Now that your project contains a data source, you can use the Visual InterDev Design-time ActiveX Control to create an Active Server Page that dynamically creates and displays a report based on this data source.

- 1 Choose the New command on the File menu. The New dialog box appears.
- 2 On the Files Tab of the New dialog box, select Active Server Page.
- 3 Enter a name for the new Active Server Page in the File name text box, and click OK. The new Active Server Page is created and appears in the Microsoft Developer Studio.
- 4 Make sure the comment <!-- Insert HTML here --> is highlighted, and choose the ActiveX Control command from the Insert | Into HTML menu. The Insert ActiveX Control dialog box appears.
- 5 Click the Design-Time Tab to display design-time ActiveX controls that are registered on your system.
- 6 Highlight the CrystalReport.DTC design-time control, and click OK.
- 7 Close the Properties dialog box if it appears. The Visual InterDev Design-time ActiveX Control appears in the Microsoft Developer Studio window.
- 8 Toggle the Use existing report check box off if it is not off already.
- 9 In the Data Connection drop-down box, select the database corresponding to the data source you added to your project.
- 10 In the Record Source drop-down box, select a database table from the database.
- 11 In the Columns list box, toggle the check box on for any field you want to become columns in your report.

NOTE: If you are familiar with the SQL language, you may prefer to click the SQL option in the Source Type box and click the Builder button. The Builder button opens the Visual InterDev Query Builder, allowing you to design a SQL statement that retrieves data from your data source.

12 Select a Crystal Smart Viewer from the Viewer drop-down box.

13 Close the design-time control.

Once again, several lines of VBScript code are added to your Active Server Page. Microsoft IIS will run this code at runtime and dynamically generate a report based on your data source.

Editing Active Server Pages

Once the Visual InterDev Design-time ActiveX Control creates the code to generate and display reports on your web site, you can edit that code to customize how the Crystal Smart Viewer and the report appear in a web browser.

Customizing the Crystal Smart Viewer

The Crystal Smart Viewer/Java is a standard Java applet, while the Crystal Smart Viewer/ActiveX is an ActiveX control. Both can be customized using the <PARAM> tags that appear under the <OBJECT> tag created in your Active Server Page by the design-time control. In your Active Server Page, look for the OBJECT tag appearing below the VBScript code. For information on how to change and set parameter for either viewer, see *Configuring the Crystal Smart Viewers, Page 31*.

Modifying the Report

Although the Visual InterDev Design-time ActiveX Control provides all of the necessary VBScript code to display your reports in a web browser, you may want to customize the output of the actual report. By using the Crystal Report Engine Object Library, provided by the Crystal Report Engine Automation Server, you can change report format, grouping, selection formulas, and sort fields.

For complete documentation, refer to *Crystal Report Engine Object Library, Page 609*. In your Active Server Page, look through the VBScript code created by the design-time control. Notice that all of the VBScript code generated is server-side script, making your web site compatible with any browser (depending on the Crystal Smart Viewer you specified).

In the VBScript code, the OpenReport call accesses the report file that you selected in the design-time control, creating an instance of the Report object. A few lines below the OpenReport method, you will find the ReadRecords method. ReadRecords obtains data and generates the final report that will be displayed in the web browser. Between these two calls, you can modify the format and data displayed in the report.

As stated before, the VBScript code provided by the design-time control creates a Report object through which your reports can be manipulated. For instance, the following code sets a record selection formula for the report at runtime:

```
session("oRpt").RecordSelectionFormula = "{customer.Region} ='CA'"
```

Notice that the Report object is stored in the session object. For complete information on the session object, refer to Microsoft documentation for Active Server Pages.

Session Timeout

By default, the session object provided by Visual InterDev for your project has a timeout value of 20 minutes. When designing web sites that use the Crystal Report Engine Automation Server and the Visual InterDev Design-time ActiveX Control, though, you will not be able to edit reports using Seagate Crystal Reports for 20 minutes after you view them in a browser. Once your web site is designed and finished, this timeout period may be just fine. However, during the development process, you may want to alter the session timeout period.

The Session object has a Timeout property that controls how long the session waits before timing out. During development, add a line of code to your Active Server Pages to set the Timeout property to a much shorter timeout period, such as 1 minute. The following code demonstrates this:

```
session.Timeout = 1
```

Be sure to remove the line of code before making your web site available to other users.

Sample Web Site

Seagate Crystal Reports includes the Xtreme Mountain Bikes, Inc., sample web site using the Crystal Report Engine Automation Server and the Crystal Active Data Driver. This web site can be installed by selecting a Custom Installation when you install Seagate Crystal Reports, and toggle the Xtreme Mountain Bike check box on under the Sample Files option in the Custom Installation Options dialog box. This site will be installed, by default, in the \CRW\SAMPLE\XTREME\ASPWEB directory.

Once installed, this web site provides a complete example of an Internet or intranet site that generates and displays Inventory reports for a fictitious Mountain Bike distribution company. Simply log on as one of the company employees listed on the home page, and browse through the reports. Different options are available depending on which user you log on as. In addition, the site provides the option of viewing the source code for any Active Server Page.

NOTE: To run this sample site, you must use Microsoft Internet Explorer 3.01 or higher.

3

Configuring the Crystal Smart Viewers

What you will find in this chapter...

Crystal Smart Viewer Overview, Page 32

Crystal Smart Viewer/Java, Page 33

Crystal Smart Viewer/ActiveX, Page 34

Crystal Smart Viewer/HTML, Page 39

Crystal Smart Viewer Overview



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Seagate Crystal Reports provides several methods for viewing reports from a web browser. Actually adding a viewer to a web page by directly editing the web page, however, is not normally necessary. The Crystal Web Report Server, for instance, can detect a user's web browser type when they request a report and automatically provide an appropriate viewer. If you develop web sites using Microsoft Visual InterDev and the Crystal Design-Time ActiveX Control, on the other hand, you can select a viewer using the design-time control, and the appropriate code will be added to your site for you.

However, there may be times when you need to manually write web pages that display one of the viewers, or you may want to edit the code created by the design-time control to customize your web site. This chapter describes how to understand and work with the Crystal Smart Viewers directly in your web pages.

NOTE: *This chapter is intended as a supplement to the web design solutions presented in Crystal Web Report Server Overview, Page 10, and Building Active Web Sites, Page 25. It is assumed that you are familiar with the concepts in at least one of those two chapters.*

Printing from the Crystal Smart Viewers

When you create a report in Seagate Crystal Reports, the program analyzes the printer that is currently selected for your system to determine font size and how to size and position objects, such as field objects and text objects on the report. If the report is then printed to a printer other than the one selected when it was created, problems with font size, clipped text, and pagination may arise.

With this in mind, consider what may happen when a report is created on one machine, served over the network by a web server on a second machine, and viewed or printed from a web browser through a Crystal Smart Viewer on a third machine. If each of these machines is connected to a different printer, report formatting problems may be compounded.

Consider a report that is designed and formatted on the first machine, where printer settings are used to determine text size and the size and position of objects in the report. When the web server generates that report, the printer it is connected to may force the length and size of a font to change. However, the field and text objects will maintain a fixed size and position. Thus, generating the report on the web server may cause text to be clipped or may leave extra blank spaces between fields.

If, on the other hand, some report objects are formatted with the *Can Grow* formatting option, these objects will resize themselves as the size of the text font is resized by the new printer. Once resized, though, those objects may change the total page number of a report or the contents that appear on each page of the report.

The Crystal Smart Viewer/Java and the Crystal Smart Viewer/HTML will display the report in a web browser as it is generated by the web server, so these formatting problems may affect how reports appear to users. The Crystal Smart Viewer/Java does not allow a user to print a report from

the web browser due to restrictions in the Java language. The Crystal Smart Viewer/HTML will simply print the HTML page exactly as it appears in your web browser. However, the Crystal Smart Viewer/ActiveX allows you to print a formatted report from a web browser. As a result, an additional level of formatting problems may appear in the printed report if the machine that the web browser is running on is connected to a third printer, different from the first two.

When designing reports that will be viewed through one of the Crystal Smart Viewers, use report fonts common on all systems to prevent resizing and pagination problems. Finally, always test reports on a client machine before distributing them to users.

Crystal Smart Viewer/Java

The Crystal Smart Viewer/Java is a standard Java applet that can be placed inside an HTML page and viewed through any browser that supports Java. Netscape Navigator, version 2.0 and later, will display reports using the Crystal Smart Viewer/Java by default. The Java viewer must have access to either the *Crystal Web Report Server Overview, Page 10*, or the Report Server Active Server Page (see *Building Active Web Sites, Page 25*). The Crystal Web Report Server can be installed on any machine running Microsoft Internet Information Server (IIS), Netscape FastTrack Server, or Netscape Enterprise Server. The Report Server Active Server Page is installed with the Crystal Design-Time ActiveX Control.

NOTE: *There may be JavaScript errors encountered while using Netscape browsers 3.x when drilling down on data then clicking the Back button to return to the last view. Please click the corresponding tab for the view you want to retrieve rather than the Back button and these errors will not occur.*

Adding the Viewer to a Web Page

As a Java applet, the Crystal Smart Viewer can be added to a web page using the standard HTML tag APPLET. The name of the public class exposed by the applet is ReportViewer. Thus, the following code displays the Crystal Smart Viewer/Java:

```
<APPLET CODE="ReportViewer.class" CODEBASE="/VIEWER/Javaviewer"
WIDTH=600 HEIGHT=400>
</APPLET>
```

When you install the Crystal Web Report Server, all class files for the applet are installed in the \CRW\CRWEB\Java Viewer directory, by default.

In addition, the Crystal Smart Viewer/Java provides several optional parameters to customize the look of the viewer and to control its functionality. Apply values to these parameters using the standard PARAM tag in your HTML code.

Parameters

The Crystal Smart Viewer/Java provides the following parameters:

- **HasGroupTree**

Indicates whether or not the viewer generates a Group Tree for the report. Does not indicate whether or not the Group Tree is displayed. Use TRUE to generate a Group Tree, FALSE to prevent a Group Tree from being generated.

- **HasRefreshButton**

Indicates whether or not a Refresh button is available in the viewer to allow the user to refresh report data. Use TRUE to allow users to refresh report data, FALSE to prevent users from refreshing report data.

- **ReportName**

Specifies the report to be displayed inside the viewer. The path must be a URL on the same server as the HTML document and must appear inside quotation marks.

- **ShowGroupTree**

Indicates whether or not the Group Tree is displayed when the viewer first appears. If the HasGroupTree parameter is set to false, this parameter is ignored. If the Group Tree is hidden, the user can display it by clicking the Toggle Group Tree button in the viewer. Use TRUE to display the Group Tree, FALSE to hide the Group Tree.

Example

The following HTML code demonstrates a possible method of embedding the Crystal Smart Viewer/Java in a web page using the APPLET tag:

```
<APPLET CODE="ReportViewer.class" CODEBASE="/VIEWER/Javaviewer"
WIDTH=600 HEIGHT=400>
<PARAM NAME=ReportName VALUE="Adcont2s.rpt">
<PARAM NAME=HasGroupTree VALUE=true>
<PARAM NAME>ShowGroupTree VALUE=false>
<PARAM NAME=HasRefreshButton VALUE=false>
</APPLET>
```

This example will display the ADCONT2S.RPT report in the Crystal Smart Viewer/Java window. A Group Tree will be generated to allow Smart Navigation, but the Group Tree will be hidden initially. The viewer does not allow a user to refresh the report data.

Crystal Smart Viewer/ActiveX

The Crystal Smart Viewer/ActiveX is an ActiveX control that can be placed inside an HTML page and viewed through any browser that supports ActiveX. Microsoft Internet Explorer, version 3.02 and later, will display reports with the Crystal Smart Viewer/ActiveX by default. The ActiveX viewer must have access to either the Crystal Web Report Server or the Report Server Active Server

Page (see *Building Active Web Sites, Page 25*). The Crystal Web Report Server can be installed on any machine running Microsoft Internet Information Server (IIS), Netscape FastTrack Server, or Netscape Enterprise Server. The Report Server Active Server Page is installed with the Crystal Design-Time ActiveX Control.

NOTE: *There may be JavaScript errors encountered while using Netscape browsers 3.x when drilling down on data then clicking the Back button to return to the last view. Please click the corresponding tab for the view you want to retrieve rather than the Back button and these errors will not occur.*

AuthentiCode Certification

The Crystal Smart Viewer/ActiveX is certified by Microsoft AuthentiCode 2.0. This certification requires Microsoft Internet Explorer 3.02 or later to open the ActiveX control. If you do not have a recent version of Internet Explorer, refer to the Microsoft Web Site to upgrade or use the Crystal Smart Viewer/Java when designing your web sites.

Adding the Viewer to a Web Page

Microsoft's Internet Explorer web browser supports the OBJECT tag in HTML. This tag can be used to add the Crystal Smart Viewer/ActiveX to a web page. Use code similar to the following:

```
<OBJECT ID="CRViewer" WIDTH=100% HEIGHT=95%
        CLASSID="CLSID:C4847596-972C-11D0-9567-00A0C9273C2A">
</OBJECT>
```

When you install the Crystal Web Report Server, the Crystal Smart Viewer/ActiveX is installed in the \CRW\CRWEB\ActiveXViewer directory, by default.

NOTE: *For complete information on the OBJECT tag, refer to your Microsoft documentation.*

Downloading the Viewer from the Server

In order for a web browser to use an ActiveX control stored on the web server, the browser must be able to download the control from the server and register it locally. The CODEBASE attribute of the OBJECT tag lets you specify the location of the original ActiveX control, relative to the current page. For example:

```
<OBJECT ID="CRViewer" WIDTH=100% HEIGHT=95%
        CLASSID="CLSID:C4847596-972C-11D0-9567-00A0C9273C2A"
        CODEBASE="/viewer/ActiveXViewer/CRViewer.dll#Version=1.2.0.25">
</OBJECT>
```

The first part of the CODEBASE attribute value indicates the location and file name of the ActiveX control as a URL address relative to the current web page. The Version attribute that appears after the # symbol allows you to specify which version of the Crystal Smart Viewer/ActiveX you want to provide your users with. The current version is 1.2.0.25.

When a web browser opens this page, it first checks the CLASSID attribute to see if the control is already registered on the client system. If not, or the version number of the viewer is lower than the

current viewers, the browser uses the CODEBASE attribute to find the control and download it. Once downloaded, the control can be registered and displayed by the browser.

Parameters

The Crystal Smart Viewer/ActiveX provides several optional parameters to customize the look of the viewer and to control its functionality. Apply values to these parameters using the standard PARAM tag in your HTML code. The Crystal Smart Viewer/ActiveX provides the following parameters:

- **Can Drill Down**

Indicates whether or not a user can drill down on summary values in a drill-down report. In a drill-down report that appears in the Crystal Smart Viewer/ActiveX, the mouse pointer becomes a magnifying glass over any group or value that can be drilled down on. Double-click the group or value to display a separate Drill-Down Tab inside the viewer.

- A value of 1(TRUE) indicates the user can drill down on reports.
- A value of 0 (FALSE) indicates the user is not allowed to drill down on reports.

- **Has Animation Control**

Indicates whether or not the viewer displays an Animation Control. The Animation Control runs while a report is being generated and downloaded. Once the report completely arrives at the client web browser, the animation stops.

- A value of 1 (TRUE) displays the Animation Control.
- A value of 0 (FALSE) prevents the Animation Control from being displayed.

- **Has Close Button**

The Close button allows a user to close the current drill-down view of the report. The Has Close Button parameter indicates whether or not the Close button is available in the Crystal Smart Viewer/ActiveX.

- A value of 1 (TRUE) displays the Close button.
- A value of 0 (FALSE) prevents the Close button from being displayed.

- **Has Group Tree**

Indicates whether or not the viewer generates a Group Tree for the report. Does not indicate whether or not the Group Tree is displayed. If HasGroupTree is set to 0, ShowGroupTree will automatically be set to 0.

- A value of 1 (TRUE) generates a Group Tree.
- A value of 0 (FALSE) prevents a Group Tree from being generated.

- **Has Navigation Control**

Indicates whether or not the page navigation buttons appear in the toolbar for the viewer. The Navigation Control includes buttons for Next Page, Previous Page, First Page, and Last Page.

- A value of 1 (TRUE) displays the Navigation Control.
- A value of 0 (FALSE) prevents the Navigation Control from being displayed.

- **Has Print Button**

Indicates whether or not the user can print the report to a printer. When the user clicks the Print button, the report is sent to a printer according to the settings selected by the Standard Print dialog box. If the Has Print Button equals 0, then you can not print. For more information, see *Printing from the Crystal Smart Viewers, Page 32*.

- A value of 1 (TRUE) displays the Print button.
- A value of 0 (FALSE) prevents the Print button from being displayed.

- **Has Progress Control**

Indicates the progress of report generation. The Progress Control is a status control that appears in the Crystal Smart Viewer/ActiveX to the right of the Search control while the report is generated. The indicator shows how much of the report has been generated, and how much is left. Once the indicator fills the Progress Control, the report is displayed and the Progress Control disappears.

- A value of 1 (TRUE) displays the Progress Control.
- A value of 0 (FALSE) prevents the Progress Control from being displayed.

- **Has Refresh Button**

Indicates whether or not a Refresh button is available in the viewer to allow the user to refresh report data.

- A value of 1 (TRUE) allows users to refresh report data.
- A value of 0 (FALSE) prevents users from refreshing report data.

- **Has Search Button**

The Search control and button that appear in the Crystal Smart Viewer/ActiveX allow a user to easily search for and jump to instances of a specific value or field that appear in the report. The user enters the value of interest in the drop-down box, then clicks the Search button to find the first instance of that value. Clicking the button again finds successive instances of the value in the report.

- A value of 1 (TRUE) displays the Search controls.
- A value of 0 (FALSE) prevents the Search controls from being displayed.

- **Has Stop Button**

While a report is being generated or downloaded, the Stop button is normally available to stop the report generation process. If the Stop button is clicked, the Crystal Smart Viewer/ActiveX will display as much report data as has been generated up to that point.

- A value of 1 (TRUE) displays the Stop button.
- A value of 0 (FALSE) prevents the Stop button from being displayed.

- **Has Zoom Control**

Use the Zoom Control to switch between levels of magnification in Crystal Smart Viewer/ActiveX. With the Zoom Control, you can magnify the report up to 400% of its original size, or reduce it down to 25% to see more of the report at once.

- A value of 1 (TRUE) displays the Zoom Control.
 - A value of 0 (FALSE) prevents the Zoom Control from being displayed.
- **Report Name**
Specifies the report to be displayed inside the viewer including the path. This is a complete URL address for the report file.
 - **Show Group Tree**
Indicates whether or not the Group Tree is displayed when the viewer first appears. If the **Has Group Tree** parameter is set to false, this parameter is ignored. If the Group Tree is hidden, the user can display it by clicking the Toggle Group Tree button in the viewer.
 - A value of 1 (TRUE) displays the Group Tree.
 - A value of 0 (FALSE) hides the Group Tree.
 - **Show Toolbar**
When Show Toolbar is set to 0, all controls in the toolbar, with the exception of the Progress Control and the Animation Control, are unavailable.
 - A value of 1 (TRUE) displays the toolbar.
 - A value of 0 (FALSE) makes the toolbar unavailable.

NOTE: If all buttons are not displayed, the toolbar disappears automatically.

Example

The following HTML code demonstrates a possible method of embedding the Crystal Smart Viewer/ActiveX in a web page using the OBJECT tag:

```
<OBJECT ID="CRViewer" WIDTH=100% HEIGHT=95%
        CLASSID="CLSID:C4847596-972C-11D0-9567-00A0C9273C2A"
        CODEBASE="/viewer/ActiveXViewer/CRViewer.dll#Version=1.2.0.25>
<PARAM NAME="Report Name"
        VALUE="http://webserver/crweb/craze/arcord1p.rpt">
<PARAM NAME="Show Group Tree" VALUE=1>
<PARAM NAME="Show Toolbar" VALUE=1>
<PARAM NAME="Has Group Tree" VALUE=1>
<PARAM NAME="Has Navigation Control" VALUE=1>
<PARAM NAME="Has Stop Button" VALUE=1>
<PARAM NAME="Has Print Button" VALUE=1>
<PARAM NAME="Has Zoom Control" VALUE=1>
<PARAM NAME="Has Close Button" VALUE=1>
<PARAM NAME="Has Progress Control" VALUE=1>
<PARAM NAME="Has Search Button" VALUE=1>
<PARAM NAME="Has Refresh Button" VALUE=1>
<PARAM NAME="Can Drill Down" VALUE=1>
<PARAM NAME="Has Animation Control" VALUE=1>
</OBJECT>
```

This example will display the ARCORD1P.RPT report in the Crystal Smart Viewer/ActiveX window. A Group Tree will be generated and displayed to allow Smart Navigation. The user can drill down on summary reports, refresh report data, and print the report to a printer.

Crystal Smart Viewer/HTML

The Crystal Smart Viewer/HTML is not an actual viewer component that can be configured. Instead, the Crystal Web Report Server or the Crystal Report Engine Automation Server translates a report into a set of web pages using HTML 3.2 format. Web browsers with support for HTML 3.2 table tags will be able to view such documents (i.e., Netscape Navigator 2.0 or later, Microsoft Internet Explorer 2.0 or later). Browsers supporting font colors and table cell background colors will render a more accurate view of actual report objects (Netscape Navigator 3.0 or later and Internet Explorer 2.0 or later).

NOTE: There may be JavaScript errors encountered while using Netscape browsers 3.x when drilling down on data then clicking the Back button to return to the last view. Please click the corresponding tab for the view you want to retrieve rather than the Back button and these errors will not occur.

Limitations of HTML Reports

Since HTML 3.2 format does not provide all of the formatting features available in the Seagate Crystal Reports report format, translating reports to HTML introduces several limitations described here. As a web administrator or designer, keep these limitations in mind when deciding how to distribute reports over your Internet or intranet site.

Object Layout/Positioning

HTML 3.2 translation preserves *relative* positioning of objects and fields. However their *absolute* positioning, height, and width is browser dependent.

Objects Translated

Object	Translated/Not Translated
Field Objects	Yes
Text Objects	Yes
Graphic, Blob, Chart Objects	Yes, as JPEG images
OLE Objects	Yes, as JPEG images
Cross-Tab Objects	Yes
Subreport Objects	Yes
Line and Box Objects	No

Limitations

Overlaid Report Objects

- HTML 3.2 does not support overlaying. Report objects which are partially overlaid (even a tiny fraction) will appear alongside each other.

Report Object Borders

- If all 4 sides of the object border are chosen, an HTML box is drawn around the report object.
- If either a bottom or top side of the object border is chosen, an HTML horizontal rule is drawn above or below the object accordingly. (Lone vertical borders are not translated.)
- Dotted lines appear as solid lines.
- Double lines appear as thick solid lines.
- Drop shadows appear as a box drawn around the report object.
- If the *Tight Horizontal* option is chosen, HTML box width will be the approximate “width of report object” or “width of data.”
- If the *Tight Horizontal* option is not chosen, HTML horizontal rule width will be the “width of report object.”
- Border/rule heights appear as “Height of font” only.

Drill-down

- Group drill-down is supported.
- Chart drill-down is not supported.

Part 2 - Crystal Report Engine Development

4

Crystal Report Engine

What you will find in this chapter...

Introduction to the Crystal Report Engine, Page 44

Before using the Crystal Report Engine in your application, Page 45

Using the Crystal Report Engine, Page 46

Crystal Report Engine API, Page 48

Exporting reports, Page 68

Handling Preview Window Events, Page 71

Distributing Crystal Report Engine Applications, Page 76

Additional Sources of Information, Page 76

Introduction to the Crystal Report Engine

Besides acting as a powerful stand-alone report creation application, Seagate Crystal Reports provides a report writing module that you can add to your own applications. As a developer using C, C++, Visual Basic, ObjectVision, Turbo Pascal, Visual dBASE, Delphi, or any programming language that can access a DLL, you can add sophisticated report generating and printing capabilities to your applications without the time-consuming task of writing your own code.

The Crystal Report Engine is a Dynamic Link Library (DLL) that allows your applications to access the same powerful report printing features that are available in Seagate Crystal Reports. As a licensed user of Seagate Crystal Reports, you receive royalty-free rights to ship the Crystal Report Engine DLL (CRPE.DLL or CRPE32.DLL) and all of its support files with any application you create.

NOTE: For more information regarding current runtime file requirements, see the Runtime File Requirements online Help.

From your application, you access the Crystal Report Engine through any of several Crystal Report Engine development tools:

- *Crystal ActiveX Control, Page 82* (CRYSTAL.OCX or CRYSTL32.OCX)
- *Crystal Report Engine Automation Server, Page 85* (CPEAUT.DLL or CPEAUT32.DLL)
- *The Crystal Visual Component Library, Page 108* (UCRPE.DCU or UCRPE32.DCU)
- *The Crystal Report Engine Class Library, Page 110* (PEPLUS.H and PEPLUS.CPP)
- *The Crystal NewEra Class Library, Page 112*
- *Crystal Report Engine API, Page 48* (CRPE.DLL or CRPE32.DLL)

When your application runs, it links with the Crystal Report Engine to access report writing functionality. Reporting can be simple, producing only a single report that is sent to a printer or preview window with no options available to the user, or it can be complex, allowing the user to change such things as record selection, sorting, grouping, or export options.

NOTE: All references to CRPE32.DLL are for the 32-bit version. If you plan on using the 16-bit version, it is called CRPE.DLL.

Sample applications

Seagate Crystal Reports comes with a number of sample applications that show you how to incorporate the capabilities of the Crystal Report Engine. Use these applications to further your understanding of the Crystal Report Engine and how to use it in various programming environments.

Special features of the Crystal Report Engine

The Development Tools included with Seagate Crystal Reports include data access and report export features that you may want to take a closer look at. Almost all developers using the Crystal Report Engine in their applications will benefit from this power.

SQL and ODBC

The Crystal Report Engine is fully compatible with most popular SQL DBMS applications, including Sybase SQL Server, Oracle, Gupta SQLBase, and Microsoft SQL Server. The Crystal Report Engine includes options for logging on to and off of SQL servers and ODBC data sources and also includes the ability to edit the SQL statement passed through to an SQL or ODBC database.

Exporting

The Crystal Report Engine enables you to print to a printer or a preview window with simple function calls. In addition, you can export a file:

- through e-mail to another person or group of people,
- directly to disk,
- to HTML for updating a web site,
- to a Microsoft Exchange folder,
- to a Lotus Notes folder, or
- to an ODBC data source.

The report can be exported in any of several word processing, spreadsheet, database file, or data exchange formats including HTML.

Before using the Crystal Report Engine in your application

Before you add the Crystal Report Engine to your application, you should be familiar with some key features of the Crystal Report Engine. Review the following points, and make sure you understand each before attempting to make calls to the Crystal Report Engine from your application.

- The Crystal Report Engine outputs existing reports. You can not create report files using the functionality of the Crystal Report Engine. Reports must be created using the Seagate Crystal Reports application described in the Seagate Crystal Reports User's Guide. Make sure you understand the report creation process before trying to print reports with the Crystal Report Engine.

NOTE: Visual Basic programmers can use the Active Data Driver, along with the Crystal Report Engine API or the Crystal Report Engine Automation Server to create reports dynamically at runtime. For more information, refer to Crystal Active Data Driver, Page 93.

- The Crystal Report Engine provides a convenient add-on to your existing application development project. With just a few lines of code, you can produce a powerful report writing and distribution tool that would take thousands of lines of code and weeks to produce otherwise.
- The Crystal Report Engine does not require the use of a fixed user interface. The Crystal Report Engine is designed to work with your existing development project and allows you to define the user interface your customers and users are familiar with and expect from your application.

Using the Crystal Report Engine

Any development project that incorporates the Crystal Report Engine requires three steps:

1. Create the reports your users will access.
2. Design the user interface that will drive the Crystal Report Engine.
3. Add the Crystal Report Engine to your application.

Creating reports

Creating reports to include with your applications is identical to creating reports for your own use; there are no restrictions. Using the procedures outlined in the Seagate Crystal Reports User's Guide and Seagate Crystal Reports online Help, create as many kinds of reports as you want to make available to your users. You can make the reports as simple or as sophisticated as your needs dictate.

While designing reports, though, keep in mind their ultimate destination. Some export formats do not support all of the formatting options available in Seagate Crystal Reports. For example, if you will be exporting reports to HTML to automatically update a web site, HTML may not support all of the fonts available on your system. This is not a limit of the Crystal Report Engine export functionality, but a limit of the HTML format itself.

If you are a Visual Basic programmer or you are using any development environment that supports Automation Servers, you may want to have reports dynamically designed for you at runtime using the Active data driver. For complete information on using the Active data driver, see *Crystal Active Data Driver, Page 93*.

Visual Basic programmers can also take advantage of the Visual Basic data control or the TrueGrid ActiveX control at runtime to dynamically produce report files. See *Using Grid Controls with the Crystal Report Engine, Page 102*, for information on using these controls with the Crystal Report Engine.

Creating the interface for printing the reports

The interface you develop to allow users to print reports is limited only by your needs and your imagination. The kind of user interface you select is unimportant to the Crystal Report Engine.

Common methods of using the Crystal Report Engine include a single menu command that produces a single report, a dialog box allowing several options for printing reports, or a completely separate front-end application that is called by your application. All are acceptable techniques, and each has its advantages. How you design your user interface can depend on any or all of the following:

- The purpose of your application.
- The types of reports your application will use.
- The printing options you want to make available with those reports.
- Whether your application will offer only one report or a choice of several reports.

Consider your application and your reporting needs carefully, and design a User Interface that will use the Crystal Report Engine most efficiently.

Adding the Crystal Report Engine to your application

As described earlier in this chapter, there are several different Crystal Report Engine development tools that can be used to add the Crystal Report Engine to your application:

- *Crystal ActiveX Control, Page 82*
- *Crystal Report Engine Automation Server, Page 85*
- *The Crystal Visual Component Library, Page 108*
- *The Crystal Report Engine Class Library, Page 110*
- *The Crystal NewEra Class Library, Page 112*
- *Crystal Report Engine API, Page 48*

Be aware that you can not use two or more of these tools in the same application.

For example, you can not create a Visual Basic application that contains the Crystal ActiveX control and also makes calls to the functions in the Crystal Report Engine API. You must choose one tool to implement the Crystal Report Engine in your project and stick with that tool.

When choosing a Crystal Report Engine tool, consider the following:

- What is your development environment?
- What is your programming ability?
- Do you need to implement the entire Crystal Report Engine or just a few features of it?

For example, the Crystal Class Library for NewEra is specifically designed for Informix NewEra. Therefore, if you are programming in Visual Basic, the Crystal Class Library for NewEra is not an option. The Crystal Report Engine Class Library, on the other hand, is based on the Microsoft Foundation Class Library for C++. To use the Crystal Report Engine Class Library, you must be using a C++ development tool, and you must be using the MFC library.

If you are an experienced programmer, you might consider the Crystal Report Engine API or the Crystal Report Engine Class Library. Novice programmers, on the other hand, may want to take advantage of the easy-to-use features of the Crystal ActiveX control, or the Visual Component Library.

The Crystal Report Engine API consists of a large number of functions exposed directly from the Crystal Report Engine DLL. These functions provide a wide range of power and flexibility for adding report writing features to your own applications. The rest of this chapter discusses the process required to use the Crystal Report Engine API in your own applications.

Although the examples in the following sections concentrate on the C programming language, the concepts should be studied by anyone using the API functions in any language. Additional information specific to Visual Basic programmers using the API can be found in *Using the Crystal Report Engine API in Visual Basic, Page 78*. Additional information for Delphi programmers is located in *Using the Crystal Report Engine API in Delphi, Page 108*. If you wish to use a Crystal Report Engine development tool other than the Crystal Report Engine API, refer to the table of contents for this manual, or search for the name of the programming language or development environment you are using in Developer's online Help.

Crystal Report Engine API

The Crystal Report Engine API (REAPI) is the most direct method of adding the Crystal Report Engine to your application project. The Crystal Report Engine itself is a Dynamic Link Library (DLL), and, therefore, exports its functionality in the form of DLL functions. These functions make up the Crystal Report Engine API.

The Crystal Report Engine DLL, CRPE.DLL (16-bit) or CRPE32.DLL (32-bit), was installed in your \WINDOWS\SYSTEM directory when you installed Seagate Crystal Reports. This assures that the DLL is available to any application on your system that uses the Crystal Report Engine.

NOTE: For complete information on distributing Crystal Report Engine and other runtime DLLs with your application, refer to the Runtime File Requirements online Help.

The process for loading a DLL and calling DLL functions is a well documented aspect of the Windows API. If you are not familiar with working with DLLs, please refer to Windows API documentation before attempting to use the Crystal Report Engine API. You may also want to consider one of the other methods described in this section for adding the Crystal Report Engine to your application.

The rest of this section assumes an understanding of DLLs and how to use them in a Windows application. It also assumes a basic understanding of the C language. The examples here are written in C, and do not cover the LoadLibrary, GetProcAddress, or FreeLibrary calls.

Many Windows development environments support direct calls to DLL functions, Visual Basic, Visual dBASE, and Delphi, for example. Refer to the documentation for your development environment for complete instructions on using a DLL. Your documentation may also cover instructions on how to translate C function calls to the language you use. Study your documentation, then review the steps described here for using the Crystal Report Engine in an application via the Crystal REAPI.

Declarations for the Crystal Report Engine API (REAPI)

Seagate Crystal Reports provides several source code files that declare the functions in the Crystal REAPI for several popular development languages. These files were installed in the Seagate Crystal Reports directory (\CRW by default) and are ready to be immediately added to your project. The following Crystal REAPI declaration files are available:

- CRPE.H declares all Crystal Report Engine API functions for C/C++.
- GLOBAL.BAS and GLOBAL32.BAS declare all Crystal Report Engine API functions for Visual Basic. For more information on using the Crystal Report Engine API with Visual Basic, see *Using the Crystal Report Engine API in Visual Basic, Page 78*.
- CRPEDB.H declares several Crystal Report Engine functions for Visual dBASE. Because of limits in the dBASE language, not all Crystal Report Engine functions are available to dBASE programmers. Refer to the individual function in Developer's online Help for information on dBASE availability.
- CRPE.PAS and CRPE32.PAS declare all Crystal Report Engine API functions for Delphi. For more information on using the Crystal Report Engine API with Delphi, see *Using the Crystal Report Engine API in Delphi, Page 108*.

NOTE: You can declare functions yourself on an individual basis, but unless you will only be using a few of the Crystal Report Engine functions in your code, it is easiest to simply copy one of the previously mentioned code files into your project directory and add it to your project.

Using the Crystal Report Engine API

The Crystal REAPI provides two options for processing and producing reports from within an application:

1. Print-Only link
2. Custom-Print link

The Print-Only link is the fastest, easiest method for producing a report with the Crystal REAPI. A Print-Only link, however, provides a very limited functionality. It allows a report to be printed on a default printer or previewed in a window on-screen. It does not allow you to customize a report in any way before printing it, though.

A Custom-Print link, on the other hand, provides all the report processing power of Seagate Crystal Reports itself. By coding a Custom-Print link into your application, you can change record selection, record sorting, group creation, group selecting, group sorting, exporting to disk files, e-mail, Exchange and Lotus Notes folders, ODBC data sources, selecting specific printers for printing, logging on to SQL servers and ODBC data sources, editing formulas, formatting report sections, and much more. A Custom-Print link is, however, a more complex process to code than a Print-Only link.

The first time you use the Crystal REAPI in your application project, you may want to start by coding a simple Print-Only link to produce basic reporting functionality. As your project develops and you become more familiar with the Crystal REAPI, you can expand the reporting functionality with a Custom-Print link.

Establishing A Print-Only Link

A Print-Only link is performed using the PEPrintReport function. The PEPrintReport function provides basic report printing functionality and demonstrates basic techniques for calling Crystal Report Engine functions from your application.

PEPrintReport enables your application to print a report, to select the output device, either a default printer or a preview window, and to specify the size and location of the preview window if the report is printed to a window. This function does not enable you to customize the report (select the records to print, set the sort order, etc.) at print time. You can set those parameters at report design time (using the Seagate Crystal Reports Design Tab), but you can not change them at print time through a Print-Only link.

If the report is sent to a preview window, you should also use the PEOpenEngine and PECloseEngine functions with your Print-Only link. PEOpenEngine and PECloseEngine allow you to control how long the preview window remains open. The window will remain open until the PECloseEngine function is called or the user clicks Close in the window. If PEOpenEngine and PECloseEngine are not used, and the report is sent to a preview window, the window will automatically close as soon as the report finishes processing.

NOTE: You may also want to get in the habit of using PEOpenEngine and PECloseEngine in all Print-Only links, as they are required steps to coding a Custom-Print link. If your code includes these functions when you design a Print-Only link, advancing the application to use a Custom-Print link in the future will be much easier.

PEPrintReport Arguments

PEPrintReport is declared in CRPE.H as follows:

```
short FAR PASCAL PEPrintReport (
    char FAR *reportFilePath,
    BOOL toDefaultPrinter,
    BOOL toWindow, char FAR *title,
    int left, int top,
    int width, int height,
    DWORD style, HWND parentWindow);
```

The following table describes each argument:

<i>Parameter</i>	<i>Description</i>
reportFilePath	The name of the report to be printed. Include the path if the report is not in the current directory. The report name can be hard-coded and unchangeable at runtime, or you can pass a string variable or character array as the result of a user choice.

<i>Parameter</i>	<i>Description</i>
toDefaultPrinter	If toDefaultPrinter is set to TRUE (1), the report is sent to a printer. The toWindow argument should be set to FALSE.
toWindow	If toWindow is set to TRUE (1), the report is sent to a preview window. The toDefaultPrinter argument should be set to FALSE.
title	The title that you want to appear in the window title bar. This argument can receive a string variable or a character array at runtime.
left	The position, in current screen coordinates, at which you want the left edge of the preview window to appear if the report is being printed to a window. Current screen coordinate measurements can be set within your application.
top	The position, in current screen coordinates, at which you want the top edge of the preview window to appear if the report is being printed to a window. Current screen coordinate measurements can be set within your application
width	The width, in current screen coordinates, of your preview window, if the report is being printed to a window. Current screen coordinate measurements can be set within your application
height	The height, in current screen coordinates, of your preview window, if the report is being printed to a window. Current screen coordinate measurements can be set within your application
style	The style setting (as defined in WINDOWS.H). Style settings can be combined using the bitwise OR operator. These are standard Windows styles. Refer to Windows API documentation for complete information on window styles. Use 0 for default style settings.
parentWindow	Specifies the window handle for the parent window to be used for this preview window.

When designing a Print-Only link using PEPrintReport, keep the following points in mind:

- If toDefaultPrinter = True, and if you have specified a printer in the report using the Printer Setup command, PEPrintReport prints to the specified printer. Otherwise it prints to the Windows default printer. If you wish to override both the printer specified in the report and the Windows default printer, you will need to establish a Custom-Print link and specify the printer using the PESelectPrinter function.

- If toDefaultPrinter = True, you may enter null values for all of the remaining parameters except reportFilePath because they apply to printing to a preview window only. The title parameter requires a null string (i.e., ""), while the rest of the parameters will accept 0 (zero).
- If parentWindow is null, Seagate Crystal Reports creates a top level window. The top left corner specified is relative to the origin of the screen.
- If parentWindow is the handle of an MDI frame window, Seagate Crystal Reports creates a preview window that is an MDI child window with the top left corner relative to the origin of the frame window's client area.
- If parentWindow is the handle of some other window, Seagate Crystal Reports creates a preview window that is a child of that window with the top left corner specified relative to the origin of the parent window's client area.
- You can use the Windows constant CW_USEDEFAULT (-32768) as the value of *left*, *top*, *width*, and *height* to indicate a default position for the preview window.

If the preview window is a top-level window and the window style is defined as 0 (i.e., the final two parameters in the PEPrintReport call are 0, 0) or, if the preview window is an MDI child window and the window style is defined as 0, Seagate Crystal Reports uses the following default style:

```
(WS_VISIBLE | WS_THICKFRAME | WS_SYSMENU | WS_MAXIMIZEBOX |
WS_MINIMIZEBOX)
```

That is, the default window is a visible window with a thick frame that can be used for sizing the window. The window includes a system menu box, and maximize and minimize buttons.

Example code for a Print-Only link

The first step in actually accessing the Crystal Report Engine is to load it into memory. This can be done just before PEPrintReport is called, when a dialog box that allows printing opens, or even when your application first starts.

Once the Crystal Report Engine is open, PEPrintReport can be called as a result of some user action, such as clicking a button on screen, or some internal application procedure.

Finally, once you are finished with the Crystal Report Engine, close it by calling PECloseEngine. If you have several print jobs, do not close the Crystal Report Engine until all print jobs are finished. Opening and closing the Crystal Report Engine uses processor time and should only be performed when necessary.

The following C code demonstrates a possible message handler for an application that provides Print-Only link functionality through a button in a dialog box. Use this code as an example of how to perform a Print-Only link.

```

short result;

switch (message)
{
    case WM_INITDIALOG:
        if (!PEOpenEngine())
            ; // Handle error
        return TRUE;

    case WM_DESTROY:
        PECloseEngine();
        return TRUE;

    case WM_COMMAND:
        switch (wParam)
        {
            case IDC_PRINTBUTTON:
                result = PEPrintReport (
                    "boxoffic.rpt",
                    FALSE, TRUE,
                    "My Report",
                    CW_USEDEFAULT,
                    CW_USEDEFAULT,
                    CW_USEDEFAULT,
                    CW_USEDEFAULT,
                    hwndParent);
                if (result != 0)
                    return FALSE;
                return TRUE;
        }
        break;
}

```

Establishing a Custom-Print link

A more advanced, and more powerful, method of using the Crystal Report Engine is through a Custom-Print link. Establishing a Custom-Print link gives you a great deal of control over your reports at runtime. For example, you can:

- set or modify the report sort order,
- set or modify the record selection and/or group selection formulas,
- modify existing report formulas,
- set or modify the database location,
- capture and evaluate Crystal Report Engine errors,

- export a report to a file, e-mail, Exchange or Lotus Notes folder, or ODBC data source,
- log on to SQL servers and ODBC data sources,
- format report sections,
- and much more.

NOTE: The Crystal Report Engine allows you to add a selection formula and sort fields to a report at runtime, even if none existed in the report when it was designed. Report formulas created in the Seagate Crystal Reports Formula Editor, however, must be added when the report is created in Seagate Crystal Reports. A formula can be edited with the Crystal Report Engine, but can not be added to an existing report from the Crystal Report Engine. Design your reports carefully, and keep this in mind when you create your application.

Coding a Custom-Print link

There are six required steps to coding a Custom-Print link in your application. Each uses a different REAPI function. The steps are:

1. *Open the Crystal Report Engine, Page 55 (PEOpenEngine).*
2. *Open a print job, Page 55 (PEOpenPrintJob).*
3. *Set the output destination, Page 55 (PEOutputToPrinter, PEOutputToWindow, or PEExportTo).*
4. *Start the print job, Page 56 (PEStartPrintJob).*
5. *Close the print job, Page 57 (PEClosePrintJob).*
6. *Close the Crystal Report Engine, Page 57 (PECloseEngine).*

In addition to these six steps, you can add several optional tasks any time after Step 2, opening the print job, and before Step 4, starting the print job. These optional tasks include changing selection formulas, editing report formulas, selecting export options, and sorting report fields.

Some REAPI functions can be called at special times to retrieve information about the print job or Crystal Report Engine. For example, PEGetVersion retrieves the current version of the Crystal Report Engine being used and can be called at any time, even without the Crystal Report Engine being open. Another example, PEGetJobStatus, can be called after Step 4 to obtain information about the current status of a job being printed. For more information on all REAPI functions, see *Crystal Report Engine Reference, Page 115*, or search for functions by name in Developer's online Help.

NOTE: The steps described here apply to a single print job. It is possible to have more than one print job open at once.

1. Open the Crystal Report Engine

Example

```
PEOpenEngine();
```

Description

This step starts the Crystal Report Engine and prepares it to accept a print job. The Crystal Report Engine must be open before a print job can be established. You should open the Crystal Report Engine before the user has a chance to try to print a report. For example, if your application uses a dialog box as the user interface to the Crystal Report Engine, open the Crystal Report Engine immediately after the dialog box is created at runtime. Your dialog box can allow the user to establish a print job and make changes to the report while the Crystal Report Engine is already open.

Every time the Crystal Report Engine is opened, it should be closed once your application is finished accessing it (*6. Close the Crystal Report Engine, Page 57*). For example, if you open the Crystal Report Engine when a dialog box is created, close the Crystal Report Engine when that dialog box is destroyed.

2. Open a print job

Example

```
job = PEOpenPrintJob( "BOXOFFIC.RPT" );
```

Description

When you open a print job, the Crystal Report Engine returns a Job Handle for the print job. This handle is important to identifying the print job in the rest of your code.

To establish a print job, PEOpenPrintJob requires the path and name of the report that is to be printed. This argument can be hard-coded into the function call, as in the example above, or you can prompt the user to choose a report for printing and pass a variable argument to the function.

To close a print job, refer to *5. Close the print job, Page 57*. In most cases, you should open the print job immediately before printing and close the print job as soon as the job is finished and the preview window is closed or printing is complete.

3. Set the output destination

Example

```
PEOutputToWindow (job, ReportTitle, CW_USEDEFAULT,  
CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, 0, NULL);
```

Description

The Crystal Report Engine must know where to send the final report. The report can be printed to a printer, displayed in a preview window, exported to a disk file, exported to another database, or exported to an e-mail address. The example above sends the report to the preview window.

Although you can choose any of the several destinations for report output, you must establish a destination for the report to print. You can, however, write code in your application that allows your users to decide on a destination themselves.

NOTE: This step does not actually print the report, it only establishes a destination for the report when printed. The report is actually printed in Step 4 using the PEStartPrintJob function.

The following functions are available to establish a print destination:

- **PEOutputToWindow**

Printing a report to a window requires no other print destination code other than the function itself.

- **PEOutputToPrinter**

Printing a report to a printer requires no other print destination code other than the function itself. However, PESelectPrinter can be used to select a printer other than the default printer at runtime. The PESelectPrinter function uses the Windows structure *DEVMODE*, *Page 353*. For more information on this structure, refer to the Windows SDK.

- **PEExportTo**

The PEExportTo function works with the PEExportOptions structure and several DLLs that control a report's export destination and format. The information required by PEExportTo can be set in your code at design time or it can work with options in your application to allow a user to specify export destination and format. If you would like to allow your users to set the destination and format of a report file, but you do not wish to program the interface to do this, use the PEGetExportOptions function to have the Crystal Report Engine provide dialog boxes that query the user for export information at runtime.

4. Start the print job

Example

```
PEStartPrintJob(job, TRUE);
```

Description

This function actually sends the report to the output device indicated in Step 3. Once PEStartPrintJob is called, the Crystal Report Engine begins generating the report. The Crystal Report Engine displays a dialog box that indicates the status of the report being generated. If the report is sent to the preview window, the window will appear as soon as PEStartPrintJob is called. The preview window can be closed by a call to PECloseWindow, by closing the Crystal Report Engine (as in Step 6), or by the user clicking the *Close* button.

As a general rule, you should avoid making any formatting changes to a print job once you call PEStartPrintJob, especially if the report is being displayed in a preview window (via PEOOutputToWindow). Formatting changes made to a report while it is still being generated and displayed in the preview window may produce undesired results, and can cause errors in the Crystal Report Engine.

5. Close the print job

Example

```
PEClosePrintJob(job);
```

Description

Once the print job has completed, it can be closed using PEClosePrintJob. If you wish to make more changes to the report and print it again, you can do so before closing the job. However, once your application is finished with a report, it should close the print job to free up memory in the user's system.

6. Close the Crystal Report Engine

Example

```
PECloseEngine();
```

Description

This function closes the Crystal Report Engine entirely. No other Crystal Report Engine functions relating to print jobs may be called once the Crystal Report Engine is closed. Therefore, you should keep the Crystal Report Engine open until it is no longer needed in your application. For example, if the Crystal Report Engine is accessed through a dialog box in your application, you should wait to close the Crystal Report Engine until the dialog box is exited and destroyed by Windows.

A Sample Custom-Print Link

The sample code below has been designed to demonstrate four of the six basic steps in establishing a Custom-print link using the C programming language. This example is based on the following scenario:

- Using Seagate Crystal Reports, you have created a report called ORDER.RPT and saved it to the C:\CRW directory. This report is a listing of customer orders, and it is the only report your application will need to print.
- In your application, you have created a Print Report menu command that opens a dialog box. The dialog box allows the user to select whether the report is printed to the printer or sent to a preview window. If the report is to be sent to the preview window, a Boolean variable called *ToWindow*, declared and initialized in another section of code not seen here, is given the value of TRUE. If the report is to just be sent straight to the printer, *ToWindow* is given the value FALSE.

- In the Print Report dialog box, there is also a *Print* button that initializes the event procedure to generate and print the report. The *Event code* section below demonstrates how the Custom-Print link can be coded in the *Print* button event procedure of your application.
- PEOpenEngine is called when the dialog box is created, and PECloseEngine is called when the dialog box is destroyed. For this reason, these two steps are not included in the custom-print link that appears below.

The topic titled *Event code* demonstrates the basic custom-print event procedure. This code includes *If* statements that check to see if an error has occurred during the call to the Crystal Report Engine. When an error occurs, you can easily handle the error in a separate routine or function. The event code below calls the function ReportError whenever an error occurs. ReportError is not a Crystal Report Engine function but is meant simply as an example of how to handle Crystal Report Engine errors. The code for ReportError appears in the section *Error code*.

Event code

```

short hJob;           /* print job handle */
BOOL bResult;

hJob = PEOpenPrintJob("C:\\\\CRW\\\\ORDER.RPT");
if (!hJob)
{
    ReportError(hJob);
    return;
}

if (ToWindow)
{
    bResult = PEOutputToWindow(hJob,
                               "My Report", CW_USEDEFAULT,
                               CW_USEDEFAULT, CW_USEDEFAULT,
                               CW_USEDEFAULT, 0, NULL);
}
else
{
    bResult = PEOutputToPrinter(hJob, 1);
}
if (!bResult)
{
    ReportError(hJob);
    PEClosePrintJob(hJob);
    return;
}
if (!PEStartPrintJob(hJob, TRUE))
{

```

```

        ReportError(hJob);
    }

PEClosePrintJob(hJob);
return;

```

Error code

```

void ReportError(short printJob)
{
    short         errorCode;
    HANDLE       textHandle;
    short         textLength;
    char          *errorText;

    errorCode = PEGetErrorCode(printJob);

    PEGetErrorText (      printJob,
                          &textHandle,
                          &textLength);

    errorText = (char*)malloc(textLength);

    PEGetHandleString(textHandle,
                      errorText,
                      textLength);

    MessageBox( hWnd, errorText,
                "Print Job Failed",
                MB_OK | MB_ICONEXCLAMATION);

    return;
}

```

Code Evaluation

Event code

The following is an evaluation of the sample event code that appears above.

```

short hJob;      /* print job handle */
BOOL bResult;

```

This section declares two local variables that are important to the remainder of the code. The variable *hJob* will receive the handle to the print job that results from a PEOpenPrintJob call. This handle is required by most Crystal Report Engine functions. *bResult* will be given a TRUE or FALSE value as the result of several Crystal Report Engine calls. Any time *bResult* receives a FALSE value, an error has occurred.

```
hJob = PEOpenPrintJob("C:\\\\CRW\\\\ORDER.RPT");
```

This call opens the new print job according to the path and file name of the report that is to be printed. In this example, the report name is hard-coded in the Crystal Report Engine call. A user would have no choice as to which report is printed. This function could also accept a character array or a pointer to a character array as an argument, allowing you to give your users the opportunity to choose a specific report for printing. PEOpenPrintJob returns a handle to the new print job, *hJob*. This handle will be used in all of the subsequent Crystal Report Engine calls shown here.

```
if (!hJob)
{
    ReportError(hJob);
    return;
}
```

This *if* statement verifies whether a valid print job handle was received in the previous line of code. If PEOpenPrintJob returned a value of 0, the print job is invalid and an error is reported. For more information on processing Crystal Report Engine errors, see the Error code section that appears below.

```
if (ToWindow)
{
    bResult = PEOutputToWindow(hJob,
                               "My Report", CW_USEDEFAULT,
                               CW_USEDEFAULT, CW_USEDEFAULT,
                               CW_USEDEFAULT, 0, NULL);
}
else
{
    bResult = PEOutputToPrinter(hJob, 1);
}
```

ToWindow acts as a Boolean variable that provides information from the user's decision as to whether this report will be printed to a preview window or to a printer. If *ToWindow* holds a TRUE value, then the user has decided to print the report to a preview window.

The *if else* code determines an output destination for the report based on the user's earlier decision. The PEOutputToWindow function prepares the Crystal Report Engine to create a preview window while PEOutputToPrinter directs the Crystal Report Engine to print the report to the default printer. (The printer used by the Crystal Report Engine can be changed with the PESelectPrinter function.) The variable *bResult* receives a FALSE value if an error occurs in either function call.

```
if (!bResult)
{
    ReportError(hJob);
    PEClosePrintJob(hJob);
```

```
    return;  
}
```

Once the appropriate destination function is called, you must verify its success and report an error if *bResult* is FALSE. ReportError is the error handling routine. It is an internal function designed to process any errors that occur during a print job. The function is passed the current value of the *hJob* handle for use in analyzing errors. Search for *Crystal Report Engine Error Codes* in Developer's online Help for information on processing errors.

NOTE: *ReportError* is not a Crystal Report Engine function, but specific to the code appearing here; it is meant only as an example of how to handle Crystal Report Engine errors.

Since a print job has been opened, you must close it after the error is reported using PEClosePrintJob. See below for more information on this function. Finally, the *if* statement causes a return after the error has been reported, thus ending the print job session.

```
if (!PEStartPrintJob(hJob, TRUE))  
{  
    ReportError(hJob);  
}
```

PEStartPrintJob actually sends the print job to the printer or a preview window. If the report is printed to a window, PEStartPrintJob creates and opens the window according to the parameters set in the PEOOutputToWindow function. If PEStartPrintJob fails (returns FALSE), an error is reported.

```
    PEClosePrintJob(hJob);
```

Once the report has printed, this print job can be closed and another one can be started if needed. If the report has been printed to a preview window, PEClosePrintJob does not close the window. The preview window is closed when the *Close* button is clicked, the PECloseWindow function is called, or the PECloseEngine function is called.

```
    return;
```

Now that the print job has finished, the event procedure can return, and the application can wait for the next user event to occur.

Error code

```
void ReportError(short printJob)  
{
```

Crystal Report Engine error processing can be most efficiently handled by a separate internal function, such as the one shown here, that is called during a print job. The Event code that is evaluated above calls the ReportError function whenever a Crystal REAPI function returns an error. The code for the ReportError function appears here as an example of how to access and evaluate Crystal Report Engine errors. The error number returned by PEGetErrorCode can be used to control how your application reacts to different types of Crystal Report Engine errors.

NOTE: The REAPI functions described here, PEGetErrorCode and PEGetErrorText, are specific to REAPI error handling. For complete descriptions of these functions, see *Crystal Report Engine Reference*, Page 115, or search for the functions by name in Developer's online Help. The function PEGetHandleString is used to retrieve variable length strings generated by different REAPI functions.

```
short      errorCode;
HANDLE     textHandle;
short      textLength;
char       *errorText;
```

Completely processing any Crystal Report Engine error requires at least four variables like those above. While only errorCode will be needed to retrieve the Crystal Report Engine error number, the other three variables will all be needed to retrieve the actual error text.

```
errorCode = PEGetErrorCode(printJob);
```

PEGetErrorCode returns a number associated with the error that has occurred. For a list of these error codes and their meanings, search for *Crystal Report Engine Error Codes* in Developer's online Help.

```
PEGetErrorText (    printJob,
                    &textHandle,
                    &textLength);
errorText = (char*)malloc(textLength);
PEGetHandleString(textHandle,
                   errorText,
                   textLength);
```

The error text must be returned in the form of a handle to a variable length string. The handle is used, along with the PEGetHandleString function to obtain the actual error text and store it in a character array. This is a complicated process, and it should be examined carefully if your code is to work.

```
MessageBox( hWnd,  errorText,
            "Print Job Failed",
            MB_OK | MB_ICONEXCLAMATION);
```

Once the error has been obtained, you can display error information to the user. This example simply opens a warning message box to alert the user of the problem. Using the error code and the error text, however, you can control Crystal Report Engine error messages any way that you find appropriate for your application.

```
return;
}
```

Once error processing is finished, you can return to processing the print job. If an error has occurred during the print job, however, then the print job should be terminated immediately after the error is processed. Review the evaluation of the event code above for ideas on how to terminate a print job after an error.

Crystal Report Engine API variable length strings

Several REAPI functions provide information in the form of a variable length string value or character array. When your program calls an REAPI function that produces a variable-length string, the Crystal Report Engine saves the string, creates a string handle which refers to the string, and returns that handle along with a value indicating the length of the string. To retrieve the contents of the string, you must call *PEGGetHandleString*, *Page 170*. This approach allows you to allocate a buffer of the exact size needed to hold the string before obtaining the actual string.

If your development language can not allocate a buffer at runtime, you should declare a reasonably large buffer. Field names and error messages will generally be less than 100 bytes, but formulas may be 1000 bytes or longer. You can control how much data is copied to the buffer when you call *PEGGetHandleString*.

Here is the procedure to follow when obtaining a variable length string:

- 1 Call the function which produces the string. This returns the string handle and length. The length includes all characters in the string plus a terminating null byte.
- 2 If necessary, allocate the string buffer.
- 3 Call *PEGGetHandleString* to copy the string from the handle into the buffer.

NOTE: PEGGetHandleString frees the memory occupied by the string handle, so you can only call this function once for a given handle.

NOTE: For experienced Windows programmers: text and name handles are Global Memory Handles for memory segments on the global heap. If you prefer, you can access these segments using the Windows GlobalLock, GlobalUnlock, and GlobalFree functions. Contents of name and text handles are null terminated ASCII strings. You must free the text handle with GlobalFree when you are done with it. (PEGGetHandleString does this for you, if you use it.)

Sample Code

Use the following C code as an example of how to call a function that returns a variable length string. The code uses the *PEGGetNthSortField* function which obtains the name of a field being used to sort the report and the direction of the sort. There are several other functions that return variable length strings, all of which are handled in a similar fashion.

Examine this code carefully and try to incorporate it into your own application without modifying the basic procedure. Only experienced programmers should try making changes to this technique since small mistakes here can cause major errors in your application. If you expect to use several REAPI functions that return variable length strings, you may want to set this code up in a separate function to avoid repetition and errors.

```
HANDLE      nameHandle;
short       nameLength;
short       direction;
char        *fieldName;
```

```

PEGetNthSortField (printJob, sortFieldN,
                   &nameHandle, &nameLength,
                   &direction);

/* allocate fieldName buffer */
fieldName = (char*)malloc(nameLength);

PEGetHandleString (      nameHandle,
                        fieldName,
                        nameLength);

/*
** fieldName now contains name
** of field and nameHandle is no
** longer valid.
*/

```

NOTE: If you retrieve a string handle but do not retrieve the string itself (i.e., you do not use *PEGetHandleString*), you should free up the string memory by calling *GlobalFree* (*nameHandle*).

Code Evaluation

```

HANDLE      nameHandle;
short       nameLength;
short       direction;
char        *fieldName;

```

Any time you evaluate a function that returns a variable length string, you will need at least three variables:

1. a handle to the string,
2. a short integer to hold the length of the string, and
3. a character array or pointer to a character array.

The direction variable in this example will hold the sort direction and is specific to *PEGetNthSortField*, *Page 200*.

It is important to note that although the *PEGetNthSortField* function is defined in the Crystal Report Engine as accepting a pointer to a handle (HANDLE*) and a pointer to a short (short*), nameHandle and nameLength are not defined as pointer variables. Instead, they are defined simply as a HANDLE and a short integer, then passed to *PEGetNthSortField* with the & operator. This technique automatically initializes the variables with the address of the variable itself. Since the *PEGetNthSortField* function requires the address in memory to place the information, this is the most convenient method to define and pass the variables.

```

PEGetNthSortField (printJob, sortFieldN,
                   &nameHandle, &nameLength,
                   &direction);

```

The PEGetNthSortField function places a handle to the sort field name in the nameHandle location and the length of the field name (all characters in the name plus a terminating null byte) in the nameLength location. These values will be used to extract the actual field name.

```
/*allocate fieldName buffer*/
fieldName = (char*)malloc(nameLength);
```

Now that you know the actual length of the field name you are trying to obtain, you can allocate exactly the right amount of memory to store that name. The malloc function does this.

NOTE: *Malloc is defined in the C runtime library stdlib.h.*

```
PEGetHandleString (      nameHandle,
                        fieldName,
                        nameLength);
```

PEGetHandleString uses the string handle to retrieve the field name and store it in *fieldName*. At the same time, *nameHandle* is invalidated. Now, the text can be used like any other character string.

NOTE: *This code is meant as a basis for your own code. Although these elements shown here are necessary for extracting a variable length string from certain Crystal Report Engine functions, experienced programmers may wish to expand the code to trap errors or handle the string text differently.*

The following is a list of all of the Crystal REAPI functions that return variable length strings:

- *PEGetAreaFormatFormula, Page 148*
- *PEGetErrorText, Page 153*
- *PEGetFormula, Page 156*
- *PEGetGroupCondition, Page 164*
- *PEGetGroupSelectionFormula, Page 168*
- *PEGetNthFormula, Page 191*
- *PEGetNthGroupSortField, Page 193*
- *PEGetNthParam, Page 195*
- *PEGetNthSortField, Page 200*
- *PEGetReportTitle, Page 212*
- *PEGetSectionFormatFormula, Page 219*
- *PEGetSelectedPrinter, Page 222*
- *PEGetSelectionFormula, Page 224*
- *PEGetSQLQuery, Page 226*

Crystal Report Engine API structures

Several REAPI functions require a structure or user-defined variable type to be passed as one or more arguments. Some of these functions require that you assign values to all members of the structure before calling the function so that the information can be used to make various settings in the Crystal Report Engine. Other functions require only the size of the structure be assigned to the StructSize member. These functions fill in the rest of the structure members for you, providing you with valuable information about a print job.

NOTE: The term structure is used here to mean both C structures and other user-defined types or records in languages such as Visual Basic and Delphi. If you are unfamiliar with this type of data, refer to the documentation for the programming language you are using.

Each structure used by REAPI is defined and explained in Developer's online Help with a link to the function that uses it. Functions that use structures also have hypertext links to the structure definitions.

Some of the structures, PEExportOptions for example, are complex, requiring other structures be passed as member values. Not all programming languages support this feature. If you are using a programming language that does not allow the use of a structure variable as a member variable defined inside other structures, declare the member variable as another data type, such as an integer or a variant data type, and assign it a value of 0 (zero) at runtime. The Crystal Report Engine will automatically provide default values or will request information from the user.

NOTE: Structure variables can not be created using Visual dBASE. Crystal Report Engine functions requiring structures as parameters are not available to dBASE.

Working with subreports

Your application can have much of the same control over subreports that it has over primary reports. The only exceptions are:

- you can not open or close a print job while a subreport is open, and
- you can only work with report sections that are actually in the subreport.

For example subreports do not have page header sections like primary reports do, so you can not do anything with a subreport that requires a page header section.

Most Crystal Report Engine functions require a print job handle as a parameter. When you supply the handle to a primary report, the functions act on the primary report. When you supply the handle to a subreport, the functions act on the subreport. Getting the handle requires a number of steps.

Opening the primary report

You must first open the primary report using the PEOpenPrintJob function. When you do this, the program returns a handle to the primary report.

Retrieving an interim subreport handle

You must then identify the subreport you want to open. You can do that using the PEGetNSubreportsInSection and PEGetNthSubreportInSection functions. When you run the PEGetNthSubreportInSection function, the Crystal Report Engine returns an interim, double word handle to the subreport you specify.

Retrieving the subreport name

Once you have the handle, you need to retrieve the name of the subreport. You do this using the PEGetSubreportInfo function. When you run this function, you pass the double word handle as the subreportHandle argument. The program retrieves the subreport name as the name member of the PESubreportInfo structure.

Opening the subreport and retrieving the job handle

Now that you have the name of the subreport (the name you assigned the subreport when you created it in Seagate Crystal Reports), you need to open the subreport. You do this using the PEOpenSubreport function. When using this function, you pass the name (or pointer to the name, depending on your development tool) as the subreportName argument. The program then opens the specified subreport and returns a job handle.

Running other Crystal Report Engine functions

Once you have the job handle, you can run any of the other Crystal Report Engine functions with the subreport, passing the subreport job handle as the printJob argument.

Changing report formats

When sending reports to a preview window using *PEOutputToWindow*, Page 257, you should always avoid making any formatting changes to a print job once you call *PEStartPrintJob*, Page 348. If the first page of a report has been displayed in the preview window, and you make formatting changes to the print job, subsequent pages of the report, if requested, may appear formatted differently than the first page. Depending on the changes made, trying to change report formatting after calling *PEStartPrintJob* can even cause errors in the Crystal Report Engine.

To avoid such formatting problems, you should get in the habit of formatting the report before starting the print job with *PEStartPrintJob*. Adding a routine to monitor job status using *PEGetJobStatus*, Page 171, can also help avoid conflicts. If you need to display the same report with different formatting options, create two separate print jobs, format each separately, and start each separately.

Exporting reports

Using the Professional Edition of Seagate Crystal Reports, you can give your applications the ability to export reports in a number of word processor and spreadsheet formats, and in a variety of popular data interchange formats as well.

The program includes two export functions, PEExportTo and PEGetExportOptions. PEExportTo can be used by itself or in conjunction with PEGetExportOptions.

- Use PEExportTo by itself if you want your application to export reports in a fixed format to a fixed destination. Use this alternative, for example, if you want to preset the format and destination for a report and have the application export the report according to your specifications in response to user input.
- Use PEExportTo in conjunction with PEGetExportOptions if you want your application to export reports in the format and destination your user selects from the Export dialog box at Print time.

PEGetExportOptions can only be used in conjunction with PEExportTo.

PEExportTo overview

The PEExportTo function uses a structure, PEExportOptions, as part of its argument list. This structure passes format and destination data to the function.

When you are using the PEExportTo function by itself, you hard code the format and destination data into the structure. Then, when you issue a PEStartPrintJob call, the program exports the report using the format and destination you specified in the code.

- Most of the format and destination data that you need to enter can be taken from the table in the PEExportTo topic.
- If you want to hard code an export file name or e-mail header information, you will have to pass a second structure as an argument to the PEExportOptions structure. This second structure is defined in the *.h file that corresponds with the destination DLL you have selected.

When you are using the PEExportTo function in conjunction with the PEGetExportOptions function, you run the PEGetExportOptions function first to:

- retrieve the format and destination data that the user specifies in the Export dialog box, and
- pass that data to the PEExportOptions structure (again, part of the PEExportTo argument list).

Then, when you issue a PEStartPrintJob call, the program exports the report using the format and destination specified by the user.

PEExportOptions structure

```
struct PEExportOptions
{
    WORD StructSize;
    // the size of the structure. Initialize to sizeof PEExportOptions
    char formatDLLName [PE_DLL_NAME_LEN];
    // Each export format is defined in a DLL. This is the name of the
    // DLL for the format you select. From table in PEExportTo topic.
    // Requires a null-terminated string. Does not need to include
    // drive, path or extension. For example, uxfsepv is an example of
    // a valid formatDLLName.
    DWORD formatType;
    // Some DLLs are used for more than one format. Enter the
    // appropriate value from the table under PEExportTo.
    void FAR *formatOptions;
    // Some formats offer additional options (see table in the
    // PEExportTo topic). You can set this element to 0. Then, If the
    // DLLs require more information, they will prompt the user
    // for it. To hard code this information, see the note immediately
    // following this structure.
    char destinationDLLName [PE_DLL_NAME_LEN];
    // Each export destination is defined in a DLL. This is the name of
    // the DLL for the destination you select. From table in PEExportTo
    // topic. Requires a null-terminated string. Does not need to
    // include drive, path or extension. For example, uxddisk is an
    // example of a valid destination DLLName.
    DWORD destinationType;
    // At the present time, each DLL implements only one destination.
    // You must specify a type here, nonetheless, because the DLL may
    // implement more than one destination someday. See the table under
    // PEExportTo for values to enter here.
    void FAR *destinationOptions;
    // Some destinations offer additional options (see table in the
    // PEExportTo topic). You can set this element to 0. Then, If the
    // DLLs require more information, they will prompt the user for
    // it. To hard code this information, see the note immediately
    // following this structure.
    WORD nFormatOptionsBytes;
    // Set by 'PEGetExportOptions', ignored by 'PEExportTo'. Both
    // functions use the same structure. PEGetExportOptions uses this
    // information in communicating with the application. The
    // application needs to know how many options bytes are being
    // returned because it may need to copy the options. PEExportTo
    // expects a filled in structure and does not need the byte
```

```

// information because it is not going to copy the options. It uses
// only a subset of the structure that does not include byte
// information.
WORD nDestinationOptionsBytes;
// Set by 'PEGetExportOptions', ignored by 'PEExportTo'. See
// comments for nFormatOptionsBytes above.
} ;

```

NOTE: You may choose to hard code the data for formatOptions and destinationOptions. You can set the formatOptions and destinationOptions elements to 0 as indicated. If the DLLs require more information than this, however, they will prompt the user to include more information. To hard code this information, you must define and fill in a structure of the appropriate kind. See the header file for the specified DLL for examples. Once the structure is defined, set the formatOptions or destinationOptions element to the address of the structure. Once PEExportTo returns or finishes, deallocate the formatOptions and destinationOptions structures. You should also deallocate the PEExportOptions structure once PEExportTo returns.

Considerations when using the export functions

The export functions are complex function calls. To avoid errors when exporting report files from your application, keep the following things in mind:

- In order to use PEExportTo and PEExportOptions, you must be using the version of the Crystal Report Engine (CRPE32.DLL) that came with the Professional Edition of Seagate Crystal Reports. If you have an earlier version of CRPE32.DLL installed on your machine and its earlier in the path, the program may find it first and not find the export functions. This can happen particularly if you are upgrading to the Professional Edition of Seagate Crystal Reports from the version of Seagate Crystal Reports that was shipped with Visual Basic Professional Edition. Visual Basic included an earlier version of CRPE32.DLL. Search your disk and delete or rename earlier versions of CRPE32.DLL, or make appropriate adjustments to your path statement.
- Make sure all format DLLs and destination DLLs are located in the same directory as CRPE32.DLL. Once Windows finds CRPE32.DLL, it will expect all of the DLL files to be in the same directory. Format DLLs are all UXF*.DLL files and Destination DLLs are all UXD*.DLL files. As a general rule, it is best to keep all of these files in the \CRW directory or the directory into which you installed Seagate Crystal Reports. Also, make certain that the PATH statement in your AUTOEXEC.BAT file includes \CRW.
- The UXF*.H and UXD*.H header files are only necessary when compiling your application. These files should be copied to the same directory as your application's source files.

Handling Preview Window Events

Using the Crystal Report Engine API, you can create a Windows CALLBACK function to handle events that occur in a preview window. For instance, if a user clicks on a button in the toolbar of the preview window, such as the Zoom button or the Next Page button, Windows registers an event for the preview window.

Using the Event functions in the Crystal REAPI, you can add instructions to your own applications to perform specific actions according to events that occur in a preview window. The sample code below demonstrates how to handle preview window events by creating a CALLBACK function for the preview window, then initializing the preview window with that CALLBACK function in your Crystal Report Engine code. The code can handle toolbar button events, Group Tree events, and even drill-down events.

The Crystal Report Engine API Event functions are only valid when a print job is sent to a preview window using PEOutputToWindow.

```
#include "crpe.h"
#include "Windows.h"

// The EventCallback function is defined as a standard
// Windows CALLBACK procedure. Return TRUE to allow the
// Crystal Report Engine to provide default behavior.
// Return FALSE to prevent default behavior from being carried out.

// The comment TODO indicates where you need to add event
// handling code specific to your application.

#if defined (WIN32)
BOOL CALLBACK EventCallback (short eventID,
                           void *param, void *userData)
#else
BOOL CALLBACK __export EventCallback (short eventID,
                           void *param, void *userData)
#endif
{
    switch(eventID)
    {
        case PE_CLOSE_PRINT_WINDOW_EVENT:
        case PE_PRINT_BUTTON_CLICKED_EVENT:
        case PE_EXPORT_BUTTON_CLICKED_EVENT:
        case PE_FIRST_PAGE_BUTTON_CLICKED_EVENT:
        case PE_PREVIOUS_PAGE_BUTTON_CLICKED_EVENT:
        case PE_NEXT_PAGE_BUTTON_CLICKED_EVENT:
        case PE_LAST_PAGE_BUTTON_CLICKED_EVENT:
        case PE_CANCEL_BUTTON_CLICKED_EVENT:
```

```

case PE_ACTIVATE_PRINT_WINDOW_EVENT:
case PE_DEACTIVATE_PRINT_WINDOW_EVENT:
case PE_PRINT_SETUP_BUTTON_CLICKED_EVENT:
case PE_REFRESH_BUTTON_CLICKED_EVENT:
{
    PEGeneralPrintWindowEventInfo * eventInfo =
        (PEGeneralPrintWindowEventInfo *) param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_GENERAL_PRINT_WINDOW_EVENT_INFO);

    // TODO
}

break;

case PE_ZOOM_LEVEL_CHANGING_EVENT:
{
    PEZoomLevelChangingEventInfo * eventInfo =
        (PEZoomLevelChangingEventInfo *) param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_ZOOM_LEVEL_CHANGING_EVENT_INFO);

    // TODO
}

break;

case PE_GROUP_TREE_BUTTON_CLICKED_EVENT:
{
    PEGroupTreeButtonClickedEventInfo * eventInfo =
        (PEGroupTreeButtonClickedEventInfo *) param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_GROUP_TREE_BUTTON_CLICKED_EVENT_INFO);

    // TODO
}

break;

case PE_CLOSE_BUTTON_CLICKED_EVENT:
{
    PECloseButtonClickedEventInfo *eventInfo =
        (PECloseButtonClickedEventInfo *)param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_CLOSE_BUTTON_CLICKED_EVENT_INFO);

    // TODO
}

break;

```

```

case PE_SEARCH_BUTTON_CLICKED_EVENT:
{
    PESearchButtonClickedEventInfo *eventInfo =
        (PESearchButtonClickedEventInfo *)param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_SEARCH_BUTTON_CLICKED_EVENT_INFO);

    // TODO
}

break;

case PE_SHOW_GROUP_EVENT:
{
    PEShowGroupEventInfo * eventInfo =
        (PEShowGroupEventInfo *)param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_SHOW_GROUP_EVENT_INFO);

    // TODO
}

break;

case PE_DRILL_ON_GROUP_EVENT:
{
    PEDrillOnGroupEventInfo * eventInfo =
        (PEDrillOnGroupEventInfo *) param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_DRILL_ON_GROUP_EVENT_INFO);

    // TODO
}

break;

case PE_DRILL_ON_DETAIL_EVENT:
{
    PEDrillOnDetailEventInfo * eventInfo =
        (PEDrillOnDetailEventInfo *) param;
    ASSERT(eventInfo != 0 && eventInfo->StructSize ==
        PE_SIZEOF_DRILL_ON_DETAIL_EVENT_INFO);

    // TODO
}

break;

case PE_READING_RECORDS_EVENT:
{

```

```

    PEReadingRecordsEventInfo * readingRecordsInfo =
        (PEReadingRecordsEventInfo *) param;
    ASSERT(readingRecordsInfo != 0 &&
           readingRecordsInfo->StructSize ==
               PE_SIZEOF_READING_RECORDS_EVENT_INFO);

    // TODO
}

break;

case PE_START_EVENT:
{
    PEStartEventInfo * startEventInfo =
        (PEStartEventInfo *) param;
    ASSERT(startEventInfo != 0 &&
           startEventInfo->StructSize ==
               PE_SIZEOF_START_EVENT_INFO);

    // TODO
}

break;

case PE_STOP_EVENT:
{
    PESTopEventInfo * stopEventInfo =
        (PEStopEventInfo *) param;
    ASSERT(stopEventInfo != 0 &&
           stopEventInfo->StructSize ==
               PE_SIZEOF_STOP_EVENT_INFO);

    // TODO
}

break;

default:
    break;
}

return TRUE;
}

// call this function after open a print job
// before call PEStartPrintJob

BOOL initializeEvent(short printJob)
{

```

```

// initialize window options
// do not have to set window options to get events,
// however, some of the events are fired only when
// certain window options are on.

PEWindowOptions windowOptions;
windowOptions.StructSize = PE_SIZEOF_WINDOW_OPTIONS;

PEGetWindowOptions(printJob, &windowOptions);

windowOptions.hasGroupTree = TRUE;
windowOptions.hasSearchButton = TRUE;
windowOptions.canDrillDown = TRUE;

if(!PESetWindowOptions(printJob, &windowOptions))
    return FALSE;

// enable event.
// by default, events are disabled.

PEEnableEventInfo eventInfo;
eventInfo.StructSize = sizeof(PEEnableEventInfo);
eventInfo.activatePrintWindowEvent = PE_UNCHANGED;
eventInfo.closePrintWindowEvent = TRUE;
eventInfo.startStopEvent = TRUE;
eventInfo.printWindowButtonEvent = PE_UNCHANGED;
eventInfo.drillEvent = TRUE;
eventInfo.readingRecordEvent = TRUE;

if(!PEEnableEvent(printJob, &eventInfo))
    return FALSE;

// set tracking cursor, gives the user feedback
// when the cursor is in the detail area
// (for a drill-down on detail event)
// use the default cursor behavior in group area.

PETrackCursorInfo cursorInfo;
cursorInfo.StructSize = sizeof(PETrackCursorInfo);
cursorInfo.groupAreaCursor = PE_UNCHANGED;
cursorInfo.groupAreaFieldCursor = PE_UNCHANGED;
cursorInfo.detailAreaCursor = PE_TC_CROSS_CURSOR;
cursorInfo.detailAreaFieldCursor = PE_TC_IBeam_CURSOR;
cursorInfo.graphCursor = PE_UNCHANGED;

if(!PESetTrackCursorInfo(printJob, &cursorInfo))
    return FALSE;

// set call back function
if (!PESetEventCallback(printJob, EventCallback, 0))

```

```
    return FALSE;  
  
    return TRUE;  
}
```

Distributing Crystal Report Engine Applications

Seagate Crystal Reports comes with a royalty-free runtime license for any application that uses the Crystal Report Engine through any of the methods described in this chapter. When distributing a Crystal Report Engine application, you must also distribute several runtime files required by the Crystal Report Engine. These files are listed in the Runtime File Requirements online Help. Be sure to carefully examine this Help file and distribute the appropriate runtime files with your application. All runtime files are included under the runtime license agreement unless otherwise stated.

Additional Sources of Information

In addition to the information provided in this chapter, you will find a wide-variety of developer topics in Developer's online Help. Many of these topics contain sample code in C, Visual dBASE, Delphi, and Visual Basic that you can copy directly into your application. For a list of all developer topics, see Developer's online Help.

If you are working with the Crystal Report Engine API in Visual Basic, refer to *Using the Crystal Report Engine API in Visual Basic, Page 78*, for information specific to Visual Basic. Delphi programmers can find information specific to using the Crystal Report Engine API with Delphi under *Using the Crystal Report Engine API in Delphi, Page 108*.

5

Visual Basic Solutions

What you will find in this chapter...

Using the Crystal Report Engine API in Visual Basic, Page 78

Crystal ActiveX Control, Page 82

Crystal Report Engine Automation Server, Page 85

Working with Active Data, Page 92

Crystal Data Object, Page 100

Using Grid Controls with the Crystal Report Engine, Page 102

Using the Crystal Report Engine API in Visual Basic

This section provides additional information for developers working in Visual Basic. Several features of the Crystal Report Engine must be handled differently in Visual Basic than in other development environments. In addition, some of the topics here are designed to simply assist Visual Basic programmers in the design of applications using the Crystal Report Engine.

When to Open/Close the Crystal Report Engine

In a Visual Basic application, you can either open the Crystal Report Engine when you open your application or when you open a form. As a general rule, it is always best to open the Crystal Report Engine when you open the application and close it when you close the application rather than when opening and closing a form. Here is why:

- When you open and close a form, the Crystal Report Engine opens every time you open the form and closes every time you close the form. If you print a report, close the form, and later decide to print a report again, the application has to reopen the Crystal Report Engine when you open the form, creating a time delay while running your application.
- When you open and close the application, the Crystal Report Engine opens once, when you start the application, and it stays open as long as the application is open. Once the Crystal Report Engine is open, you can print a report as often as you wish without the need to reopen the Crystal Report Engine every time you print.

Embedded Quotes in Visual Basic Calls to the Crystal Report Engine

When you pass a concatenated string from Visual Basic to the Crystal Report Engine (for example, for a record selection formula), it is important that the resulting string has the exact syntax that the Crystal Report Engine expects. You should pay special attention to embedded quotes in concatenated strings because they are often the source of syntax errors.

Several examples follow. The first example shows code with a common embedded quote syntax error and the last two examples show code using the correct syntax.

Incorrect syntax

```
VBNameVariable$ = "John"  
Recselct$ = "{file.LASTNAME} = " + VBNameVariable$
```

This code results in the string:

```
{file.LASTNAME} = John
```

Since John is a literal string, the Formula Checker expects to see it enclosed in quotes. Without the quotes, the syntax is incorrect.

Correct syntax

```
VBNameVariable$ = "John"  
Recselct$ = "{file.LASTNAME} = " +  
(chr$(39) + VBNameVariable + chr$(39))
```

This code results in the string:

```
{file.LASTNAME} = 'John'
```

This is the correct syntax for use in a Seagate Crystal Reports record selection formula. This is the syntax you would use if you were entering a selection formula directly into Seagate Crystal Reports.

```
VBNameVariable$ = "John"  
Recselct$ = "{file.Lastname} = "  
(+ "' " + VBNameVariable + "'")
```

This code also results in the string:

```
{file.LASTNAME} = 'John'
```

Again, the correct syntax.

Passing Dates/Date Ranges in Visual Basic using the Crystal Report Engine API Calls

You may want to pass date or date range information from your Visual Basic application to the Crystal Report Engine for use in formulas, selection formulas, etc. Here is an example showing a way to do it successfully:

- 1 Start by opening a print job and assigning the print job handle to a variable.

```
JobHandle% = PEOpenPrintJob ("C:\CRW\CUSTOMER.RPT")
```

- 2 Create variables that hold the year, month, and day for both the start of the range and the end of the range.

```
StartYear$ = 1992  
StartMonth$ = 01  
StartDay$ = 01  
EndYear$ = 1993  
EndMonth$ = 12  
EndDay$ = 31
```

- 3 Now build a string to pass to the record selection formula. This is done in two steps:

- First, build the starting and ending dates for the date range.
 - Assign the starting date string to the variable StrtSelect\$.
 - Assign the ending date string to the variable EndSelect\$.

```

StrtSelect$ = "{filea.STARTDATE} < Date
(" + StartYear$ + ", " + StartMonth$ + ", "
+ StartDay$ + ")"
EndSelect$ = "{filea.ENDDATE} < Date
(" + EndYear$ + ", " + EndMonth$ +
", " + EndDay$ + ")"

```

- Second, build the selection formula using the StrtSelect\$ and EndSelect\$ variables.

```
Recselct$ = StrtSelect$ + " AND " + EndSelect$
```

- Once your formula is built, set the record selection formula for the report.

```
RetCode% = PESetSelectionFormula
(JobHandle%, RecSelect$)
```

- Finally, print the report.

```
RetCode% = PESTartPrintJob (JobHandle, 1)
RetCode% = PEClosePrintJob (JobHandle, 1)
```

- Modify this code to fit your needs.

“Print file exists” Error Message when Printing to File in Visual Basic

When using the *PEOutputToFile*, *Page 251*, call, the Crystal Report Engine will not print to a file that already exists. If the file name you have specified as *outputFilePath* is already in use by another file, you will get a “Print file exists” error message. To avoid this, you either have to assign a different file name as *outputFilePath* or delete any file that has the same name as the *outputFilePath* prior to issuing the call. You can easily delete existing files by placing code like this prior to your *PEOutputToFile* call:

This call uses the “exists” function that is listed below:

```

If exists(FileName) Then
    Msg = FileName + " already exists.
    OK to overwrite?"
    choice = MsgBox (Msg, 36)

If choice = 6 Then ' The user said yes
    Kill FileName

' What follows is code for the exists function.
' This function returns True if a given file
' exists, False otherwise.

Function exists (f As String) As Integer
    Dim n As Integer
    On Error GoTo handler
    n = FreeFile

```

```

' Try to open file for input.
' If successful, file exists.

Open f For Input As #n
Close #n
exists = True
Exit Function

handler:

' If you get here the file does not exist.

exists = False
Exit Function

End Function

```

Identifying String Issues in Visual Basic Links to the Crystal Report Engine

When you are passing a string to the Crystal Report Engine as part of your Custom-Print link, you may think that you are passing one thing when the program, in fact, is passing something entirely different. This can happen easily, for example, when you are passing a concatenated string that you have built using variables. A small syntax error (with embedded quotes, for example) can lead to an error message and a failed call. A simple debugging procedure follows.

To Identify a String Issue (bug)

To identify a string bug, have the program display what it is passing in a message box. To do so, put a line of code similar to the following immediately after the call in question:

```
MsgBox (variablename)
```

Look at the string that is displayed and make certain that it is exactly what Seagate Crystal Reports expects for a string.

- If the syntax is incorrect, look for errors in the concatenated string you have built.
- If the syntax is correct, look for other problems that could have caused the call to fail.
- If you are not sure if the syntax is correct, write down the string from the message box, enter it in the Seagate Crystal Reports Formula Editor, and click the *Check* button. If there is an error in the string, the Formula Checker will identify it for you.

Hard-coded Nulls in Visual Basic Structures

When you assign a string to a structure in Visual Basic, it is necessary to hard-code a null immediately after the string. For example:

```
myStruct.stringField = "Hello" + CHR$(0)
```

Visual Basic Wrapper DLL

Some of the features of the Crystal Report Engine API are not directly available to Visual Basic programmers, due to restrictions in the Visual Basic language. To avoid problems calling functions in the API, you may want to consider using the *Crystal ActiveX Control, Page 82*, or the *Crystal Report Engine Automation Server, Page 85*. However, if you prefer to work with the API, Seagate Crystal Reports includes the Visual Basic Wrapper DLL, CRWRAP16.DLL (16-bit) and CRWRAP32.DLL (32-bit).

The Visual Basic Wrapper DLL has been designed specifically for programming in the Visual Basic environment and can be used to build Crystal Report Engine applications in Visual Basic 4.0 or later. The CRWRAP.BAS module, installed by default in the \CRW directory, redefines many of the functions and structures defined in GLOBAL.BAS. When working with the Crystal Report Engine API, add both modules to your Visual Basic project.

The functions and structures defined in the Visual Basic Wrapper DLL provide an interface for handling export formats, parameter fields, SQL server, and ODBC logon information, graphs, printing, and more. For complete information on each of the structures and functions included, search for *Visual Basic Wrapper for the Crystal Report Engine* in Developer's online Help. In most cases, each function or structure has a corresponding function or structure in the original Crystal Report Engine API with a similar name. When working in Visual Basic though, you must use the functions and structures provided by the Visual Basic Wrapper DLL.

Crystal ActiveX Control

ActiveX controls bring more powerful applications to desktops and networks. ActiveX moves beyond applications that produce static documents to a Windows environment that provides active controls, documents, and client applications that can operate and interact not only with each other, but also with network intranets and the global Internet.

ActiveX controls provide plug-in capabilities that let you add application components, and even entire applications, to your own development projects without writing a line of code. Seagate Crystal Reports includes the Crystal ActiveX Control. Use the ActiveX Control to easily add all of the report processing power of Seagate Crystal Reports to your own Visual Basic, Visual C++, Borland C++, Delphi, and other applications.

NOTE: Your development tools may refer to an ActiveX Control by any of the following names: OLE Control, OCX Control, Custom Control, or ActiveX Control. As long as the term used refers to a control with an .OCX filename extension, it is synonymous with the term ActiveX Control used here.

NOTE: Seagate Crystal Reports also includes a Visual Basic Custom Control (CRYSTAL.VBX). However, the ActiveX Control is based on more advanced technology and provides more features. You should use the ActiveX Control for development of any new Visual Basic projects. To upgrade an existing project to use the ActiveX Control, see Upgrading from the Crystal Custom Control, Page 85. If, for some reason, you choose to use the VBX in your project rather than the ActiveX Control, the VBX has been fully documented in Developer's online Help.

Adding the ActiveX Control to your Project

This section demonstrates how to add the Crystal ActiveX Control to an application project being designed in Visual Basic 4.0. If you wish to use the ActiveX Control in a different development environment or a more recent version of Visual Basic, please refer to the documentation that came with your development tools for information on adding an ActiveX or OLE Control (OCX) to your project.

The Crystal ActiveX Control was installed in the \WINDOWS\SYSTEM directory when you installed Seagate Crystal Reports. You add the ActiveX Control to your Visual Basic project using the Custom Controls command on the Visual Basic Tools menu.

- 1 Open Visual Basic.
- 2 Open the project to which you want to add the ActiveX Control.
- 3 Choose the Custom Controls command from the Tools menu. The Custom Controls dialog box appears.
- 4 If Crystal Report Control appears in the Available Controls list, click the check box next to it, click OK, and skip to Step 8.
- 5 If Crystal Report Control does not appear in the Available Controls list, click Browse. The Add Custom Control dialog box appears.

NOTE: Crystal Report Control is the name of the Crystal ActiveX Control when it is added to a development project. The term ActiveX Control refers to a type of control, while Crystal Report Control is the name of the ActiveX Control provided by Seagate Crystal Reports.

- 6 Use the controls in the Add Custom Control dialog box to locate and select the CRYSTL16.OCX (16-bit) or CRYSTL32.OCX (32-bit) file. This file was installed in your \WINDOWS\SYSTEM directory when you installed Seagate Crystal Reports. Once you locate and select the file, click Open.
- 7 Crystal Report Control will now appear in the Available Controls list box. Click the check box next to the name of the control, and click OK.
- 8 Visual Basic adds the Crystal ActiveX Control to your toolbox. The tool looks like this:



- 9 When you want to add the ActiveX Control to a form, double-check the tool and the program installs it on the active form.

NOTE: The ActiveX Control can be added to AUTOLOAD.MAK to automatically load the Control to your project. Refer to Visual Basic documentation for information on adding controls to AUTOLOAD.MAK.

NOTE: For instructions on how to add an ActiveX Control or OLE control to development applications other than Visual Basic, refer to the documentation that came with the development application you are using.

Using the ActiveX Control

Once you have the ActiveX Control object on your form, you build the connection between your application and Seagate Crystal Reports by setting the object's properties at design time or changing properties at runtime. The ActiveX properties let you specify:

- the name of the report you want to print in response to an application event,
- the destination for that report (window, file, or printer),
- the number of copies you want to print (if your report is going to the printer),
- print file information (if your report is going to a file),
- preview window sizing and positioning information (if your report is going to a window),
- selection formula information (if you want to limit the records in your report),
- sorting information, and
- other related properties.

Crystal ActiveX Control properties can be changed either at design time or at runtime. Note, however, some properties are available only at runtime. These properties do not appear at design time.

NOTE: For a complete description of each property in the Crystal ActiveX Control, refer to Developer's online Help.

Changing Properties for the ActiveX Control

1. Click the ActiveX control on your form to select it.
2. Right-click and choose the Properties command from the shortcut menu that appears. The Crystal Report Control Properties dialog box appears.
3. Use the tabs and controls in this dialog box to change the ActiveX Control properties at design time.

NOTE: ActiveX Control properties also appear in the Visual Basic Properties box. For instructions on using the Properties box, refer to your Visual Basic documentation.

Changing Properties at Runtime

You can set most of the ActiveX Control properties at runtime by adding simple entries to your procedure code. Runtime property settings replace settings you make via the Properties list at design time.

Use the Action property or the PrintReport method to actually process the report at runtime. The Action property and the PrintReport method can only be used at runtime, and are the only means by which a report can actually be generated by the ActiveX Control.

For information on how to set ActiveX Control properties at runtime, refer to their syntax by searching for each property by name in Developer's online Help. Included are examples of how to set each property at runtime.

Upgrading from the Crystal Custom Control

If you are using the Crystal Custom Control (CRYSTAL.VBX) in a Visual Basic project, you can upgrade your project to use the more powerful Crystal ActiveX control. All previous code and settings will be retained when you upgrade your project.

Normally, Visual Basic versions 4.0 and 5.0 automatically upgrade the control used in your project when you simply open the project in the Visual Basic environment. If Visual Basic does not upgrade your Crystal Custom Control correctly, open the VB.INI file in a text editor, such as Notepad, and verify the following settings exist in the appropriate sections.

For 32-bit environments:

```
[VBX_Conversions32]
crystal.vbx={00025600-0000-0000-C000-000000000046}
#5.0#0;c:\windows\system\crystl32.ocx
```

For 16-bit environments:

```
[VBX_Conversions16]
crystal.vbx={00025600-0000-0000-C000-000000000046}
#5.0#0;c:\windows\system\crystl16.ocx
```

NOTE: The actual path indicated should correspond to the location of the Crystal ActiveX Control. The path on your system may or may not be the same as the path shown here. In addition, each entry should appear on a single line in your VB.INI file.

Crystal Report Engine Automation Server

The Crystal Report Engine Automation Server has been designed as both an object-oriented approach to adding Crystal Report Engine features to your applications, and as an ideal method for displaying reports in web pages. If you work in a development environment that supports access to COM-based automation servers, such as Visual Basic, you will quickly make full use of the Crystal Report Engine Automation Server to add powerful reporting to your applications. In addition, if you manage a web server that supports Active Server Pages, such as Microsoft's Internet Information Server (IIS), Personal Web Server, or Peer Web Services, the Crystal Report Engine Automation Server satisfies all of your dynamic reporting needs.

Seagate Crystal Reports provides both 16-bit (CPEAUT16.DLL) and 32-bit (CPEAUT32.DLL) versions of the Crystal Report Engine Automation Server, depending on the version of Seagate Crystal Reports you installed. The Crystal Report Engine Automation Server was installed in your \WINDOWS\SYSTEM directory when you installed Seagate Crystal Reports.

The Crystal Report Engine Automation Server is an in-process automation server based on the Component Object Model (COM). This automation server provides an IDispatch interface, but is not programmable through a vtable interface. For Visual Basic programmers and in Active Server Pages, handling the IDispatch interface is almost transparent. For more information on the Component Object Model and COM interfaces, refer to Microsoft documentation.

Adding the Automation Server to your Visual Basic Project

Before you can use the Crystal Report Engine Automation Server with a Visual Basic project, it must be registered on your system, and you must add it to your project. If you selected to install Development Tools when you installed Seagate Crystal Reports, the automation server will have already been registered on your system. If you did not select Development Tools, run the Seagate Crystal Reports setup application again, select Custom installation, and make sure Development Tools are installed.

To add the automation server to a project in Visual Basic 4.0, use the following procedure:

- 1 With your project open in Visual Basic, choose the References command from the Tools menu. The References dialog box appears.

NOTE: For complete information on adding ActiveX components to a project, refer to your Visual Basic documentation.

- 2 The Available References list box shows all available component object libraries currently registered on your system. Scroll through this list box until you find the *Crystal Report Engine Object Library*. This is the Crystal Report Engine Automation Server.

NOTE: If the Crystal Report Engine Object Library does not appear in the Available References list box, use the Browse button to locate and select the Crystal Report Engine Automation Server (CPEAUT16.DLL or CPEAUT32.DLL) in your \WINDOWS\SYSTEM directory.

- 3 Toggle on the check box next to the Crystal Report Engine Object Library reference. This makes the Crystal Report Engine Automation Server available to your project.
- 4 Click OK in the References dialog box.

Using the Automation Server in Visual Basic

There are four primary steps to using the Crystal Report Engine Automation Server in your Visual Basic project:

1. *Creating an Application Object, Page 86*
2. *Obtaining a Report Object, Page 87*
3. *Using the Report Object, Page 87*
4. *Releasing Objects, Page 88*

Step 3, *Using the Report Object, Page 87*, can consist of calling and using any methods or properties available in the automation server's object library. It is the step in which you design your custom report printing functionality for your application. The rest of the steps, however, are necessary for any implementation of the object library in a project and have certain requirements.

Creating an Application Object

The Application object in the Crystal Report Engine Automation Server's object library is the only object that can be created. Using the Application object, you can obtain a report object by opening a report file, manipulate aspects of the report object, such as select formulas and sort fields, then print or export the report.

Since the Application object is the only creatable object exposed by the Crystal Report Engine Automation Server, you must create an Application object before you can perform any other tasks using the Crystal Report Engine. Use code similar to the following to create an Application object in your Visual Basic project:

```
Dim app As Object  
Set app = CreateObject("Crystal.CRPE.Application")
```

Crystal.CRPE.Application is the Application object's ProgID (programmatic identifier). Visual Basic uses this ID to create an instance of the Application object, which can then be used to obtain a Report object. For a complete description of the CreateObject function, refer to your Visual Basic documentation.

Obtaining a Report Object

You obtain a Report object by specifying a Seagate Crystal Reports (.RPT) file and opening it with the OpenReport method of the Application object:

```
Dim report As Report  
Set report = app.OpenReport("c:\reports\craze.rpt")
```

The OpenReport method has only one parameter, the path of the report file you want to access. By setting a report object according to the return value of this method, you can proceed to manipulate, print, preview, or export the report using other objects, methods, and properties available in the Crystal Report Engine Automation Server's object library.

Using the Report Object

Once you obtain a Report object, you can use that object to make runtime changes to the report file, then send the report to a printer, a preview window, a disk file, an e-mail address, an ODBC data source, or a group folder in Microsoft Exchange or Lotus Notes. Note that the changes you make at runtime are not permanent; they do not change the original report file, they only affect the output of the report during the current Crystal Report Engine session.

Through the report object, you obtain access to different aspects of the report file, such as selection formulas, subreports, sort fields, and format settings. For example, the following code changes the record selection formula for the report:

```
report.RecordSelectionFormula = "{customer.Region} = 'CA'"
```

Refer to the reference section of this manual for complete information on all objects, properties, and methods available in the object library and how to use them.

Once you make all desired changes and review settings for the report using the functionality available in the automation server, you can print, preview, or export the report just as you do from Seagate Crystal Reports. The automation server provides default settings for these activities, or you can specify your own settings. The simplest technique for sending the report to a printer would look like this:

```
report.PrintOut
```

Without receiving any parameters, the PrintOut method simply sends the report to the default printer with default settings. For more information about methods for the Report object, search for each method by name in Developer's online Help.

Releasing Objects

Visual Basic will clean up any objects that have not been released when your application terminates. However, since objects use memory and system resources that can not be accessed by other running applications, you should get into the habit of releasing any objects when you are finished with them.

To release an object, simply set it equal to Nothing:

```
Set report = Nothing  
Set app = Nothing
```

Handling Errors

Error trapping for the Crystal Report Engine Automation Server can be handled just like normal error trapping in Visual Basic. When an error occurs in the automation server, the error is sent to Visual Basic which sets the properties of the Err object appropriately. To avoid runtime problems, you should trap for errors inside your Visual Basic code. A typical error trapping routine might look something like this:

```
On Error GoTo HandleError  
' Several lines of  
' Visual Basic code  
HandleError:  
    If (Err.Number <> 0) Then  
        MsgBox (Err.Description)  
    End If
```

The advantage of handling automation server errors like this is that they can be handled at the same time other Visual Basic errors are handled, making your code more efficient and easier for other developers to understand.

Object Name Conflicts

Some object names in the Crystal Report Engine Object Library may conflict with object names in other object libraries attached to your Visual Basic projects. For instance, if your project includes the Data Access Objects (DAO) Object Library, the DAO Database object can conflict with the Report Engine Object Library's Database object. Such name conflicts can produce errors in your applications.

To avoid name conflicts, you should append all references to Crystal Report Engine Object Library object names with CRPEAuto, the name of the object library as it appears in Visual Basic. For instance, the following code can be used to create a Report object:

```
Dim rpt As CRPEAuto.Report  
Set rpt = app.OpenReport("c:\reports\craze.rpt")
```

Object names in other object libraries should also be appended with an object library name. For instance, the DAO Database object could be appended with DAO:

```
Dim db As DAO.Database
```

Viewing the Crystal Report Engine Object Library

The Visual Basic Object Browser allows you examine the classes, methods, and properties exposed by any ActiveX component available to your project. If you have selected the Crystal Report Engine Object Library using the References dialog box (see *Adding the Automation Server to your Visual Basic Project, Page 86*), then you can browse through the Object Library using the Visual Basic Object Browser:

- 1 With your project open in Visual Basic, choose the Object Browser command from the View menu. The Object Browser appears.
- 2 From the Libraries/Projects drop-down box, select the *Crystal Report Engine Object Library*. Classes, methods, and properties exposed by the Object Library will appear in the Object Browser.
- 3 Select a class in the Classes/Modules list box to view its methods and properties in the Methods/Properties list box.

NOTE: While viewing the Crystal Report Engine Object Library in the Visual Basic Object Browser, you may notice several classes, methods, and properties that are not documented in the Seagate Crystal Reports Technical Reference. There are several features in the Crystal Report Engine Automation Server that are not available with Seagate Crystal Reports, and are protected by a security feature built into the Object Library. These features will become available in future Seagate Software products. Contact Seagate Software's Sales department for further information.

Seagate Crystal Reports also provides the Crystal Report Engine Object Library Browser Application as a convenient utility for accessing online information about the Crystal Report Engine Object Library. Simply choose the Xtreme Mountain Bike option in the Sample Files when installing, or choose an automatic installation (the files will be installed by default) to install the utility, then browse through the Object Library using the tree control. Select a class, method, or property for more information on how to use it.

Handling Preview Window Events

The Report and Window objects in the Crystal Report Engine Object Library include several Events. By handling these events in your Visual Basic project, you can customize how your application responds to user actions. For instance, if a user clicks on a button in the toolbar of the preview window, such as the Zoom button or the Next Page button, your application can respond to that event.

NOTE: Events are only available in Visual Basic 5.0. If you are using a version of Visual Basic earlier than 5.0, you will not be able to make use of the Events exposed by the Report or Window object.

To handle Events for the Report or Window object, you must declare the instance of the object as **Public** and **WithEvents**. For example:

```
Public WithEvents repEvents As Report  
Public WithEvents wndEvents As Window
```

Once declared, the objects will appear in the Visual Basic Object window. If you select the object, its Events will be displayed, just as if you were working with any other Visual Basic object.

NOTE: *The Window object events are only valid when a report is sent to a preview window using the Preview method.*

When working with Report or Window objects that have been declared WithEvents, you can only set the object once using the Set command. You can not reset a Crystal Report Engine object declared WithEvents. Every time you want to access a new report with events, you must create a new Report object variable. Likewise, every time you want to display a preview window with events, you must create a new Window object variable.

The following code demonstrates how to set up and use events for both the Report object and the Window object. Actual event handling code is left for you to fill in. You are limited only by the restrictions of the Visual Basic language.

```
Option Explicit  
Public WithEvents rpt1 As Report  
Public vw1 As View  
Public WithEvents wnd1 As Window  
  
Private Sub Command1_Click()  
    Set vw1 = rpt1.Preview  
    Set wnd1 = vw1.Parent  
End Sub  
  
Private Sub Form_Load()  
    Set appl = CreateObject("Crystal.CRPE.Application")  
    Set rpt1 = appl.OpenReport("c:\crw\rt01.rpt")  
  
    rpt1.EventInfo.ActivatePrintWindowEventEnabled = True  
    rpt1.EventInfo.ClosePrintWindowEventEnabled = True  
    rpt1.EventInfo.GroupEventEnabled = True  
    rpt1.EventInfo.PrintWindowButtonEventEnabled = True  
    rpt1.EventInfo.ReadingRecordEventEnabled = True  
    rpt1.EventInfo.StartStopEventEnabled = True  
  
End Sub  
  
Private Sub rpt1_Start(ByVal Destination As _  
                      CRPEAuto.CRPrintingDestination, _  
                      useDefault As Boolean)
```

```

    ' Put event handling code here.

End Sub

Private Sub rpt1_Stop (ByVal Destination As
                      CRPEAuto.CRPPrintingDestination,
                      ByVal Status As CRPEAuto.CRPPrintingProgress)

    ' Put event handling code here.

End Sub

Private Sub wnd1_ActivatePrintWindow()

    ' Put event handling code here.

End Sub

Private Sub wnd1_ClosePrintWindow (useDefault As Boolean)

    ' Put event handling code here.

End Sub

' Other events for the Report and Window objects
' can be seen by using the Visual Basic Object Browser
' with the Crystal Report Engine Object Library.
' (a lightning bolt icon appears next to Events in
' the Object Browser.)

' Once and instance of a Report object or Window object,
' is declared, you can add Event handlers to your code by
' selecting the object in the Visual Basic Object list and
' then selecting the desired event.

```

For complete descriptions of all available Crystal Report Engine Object Library Events, refer to the *Report Object, Page 685*, and the *Report Object, Page 685*.

Distributing the Automation Server with Visual Basic Applications

When you finish designing your application and decide to distribute it to your users, you must make sure that the Crystal Report Engine Automation Server is distributed with it. In addition, you must make sure the automation server gets registered on your users' systems. The easiest way to do this is to use the Application Setup Wizard installed with Visual Basic.

This Wizard leads you through the process of designing a setup application for your project. In addition, the Setup Wizard detects any ActiveX components included in your project and leads you through the process of adding code to the setup application to include the required files and register the components on a user's machine.

For more information about files that need to be distributed with Crystal Report Engine applications, refer to Runtime File Requirements online Help.

Sample Applications

Seagate Crystal Reports includes a complete sample application written in Visual Basic 5.0 using the Crystal Report Engine Automation Server. The Xtreme Mountain Bike Inventory Application is a complete real-world application that provides various reports to employees at a fictitious company. Report access is restricted based on user logon information. The application is located in \CRW\SAMPLE\XTREME\INVNTORY and provides the option of viewing the source code for any Visual Basic form displayed.

In addition, a self-extracting executable located in the \CRW\SAMPLE directory contains three small sample applications that demonstrate various aspects of the Crystal Report Engine Automation Server. Simply run the SAM32AUT.EXE (32-bit) or SAM16AUT.EXE (16-bit) application to install the samples. The three samples are:

- **AUBASIC**
Demonstrates the basic code required to open a report and print, preview, or export it using the Crystal Report Engine Automation Server.
- **AUBROWSE**
Demonstrates how to browse through the areas of a report and access the objects in each area.
- **AUFMLA**
Demonstrates how to get and set record selection formulas, group selection formulas, and SQL queries stored with a report.

Working with Active Data

Normally, developing applications using the Crystal Report Engine requires designing and saving one or more report files in advance to be accessed by the application at runtime. This process requires that the programmer has access to the data during design time, and that the application, upon installation, also installs whatever database drivers and files are required to make sure the reports can connect to the required data.

An alternative to runtime connectivity, of course, is to save data with the report files. The data is neatly packaged and available whenever the report is requested from your custom application. However, saving data with a report increases the file size of the report, wasting disk space. Furthermore, this technique produces a static report file in which the data can not be updated without connectivity to the database.

The Crystal Active Data Driver allows you to create report files at design time without specifying an actual data source. Instead, the report is based on a field definition file, an ASCII text file with place holders to represent database fields. At runtime, you add code to your application to specify the actual source of data for the report.

The Active Data Driver supports Data Access Objects (DAO), Remote Data Objects (RDO), Active Data Objects (ADO), the Visual Basic Data Control, and Crystal Data Objects (CDO). For complete information on DAO and ADO, refer to Microsoft documentation. For information on the Data Control, refer to your Visual Basic documentation. For information on CDO, see *Crystal Data Object, Page 100*.

Crystal Active Data Driver

A report file designed with the Crystal Active Data Driver, instead of a specific database or ODBC driver, contains information about the kind of data to place in the report instead of information about an actual data source. It looks for field types, rather than actual fields. These field types are specified in a field definition file.

A field definition file is a simple ASCII text file with tab-delimited fields. Each field in the text file represents a field that must exist in the data source at runtime. For an example of a field definition file, refer to the file ORDERS.TTX installed in the \CRW directory.

At design time, you create your report based on the field definition file. Previewing or printing the report at design time has little value except to format field placement and style. Since there is no real data in the text file, you can not preview or print any data at design time.

At runtime, your application opens the report file, just as it would any other report file. Instead of simply formatting and printing the file at runtime, though, you change the data source pointed at by the Crystal Active Data Driver, which is the field definition file, to a Recordset or Rowset object for an Active data source. An Active data source can be any data source that supports the ActiveX concept. The most common Active data sources available are through Microsoft's DAO, RDO, and ADO and through the Visual Basic Data Control. In addition, this version of Seagate Crystal Reports introduces CDO (see *Crystal Data Object, Page 100*).

Once the Crystal Active Data Driver obtains the Recordset from the runtime data source, the Crystal Report Engine can generate the actual report using existing data. The entire process saves you time designing reports and produces reports that are much more flexible and portable.

Using the Active Data Driver

Designing and generating reports using the Crystal Active Data Driver is a straightforward process, but requires several specific steps:

1. *Create a Field Definition File, Page 93*
2. *Design the Report, Page 95*
3. *Obtain a Recordset from the Runtime Data Source, Page 95*
4. *Open the Report, Page 96*
5. *Pass the Recordset to the Active Data Driver, Page 97*
6. *Print the Report, Page 97*

The following sections demonstrate this process using the Crystal Active Data Driver with the Crystal Automation Server in Visual Basic 4.0.

Create a Field Definition File

A field definition file is a tab-separated text file that contains information about field names, field types, and sample field data. Field names used in the field definition file must match the field names that will appear in the active data source that is specified at runtime. Field type information indicates the type of data in each field (string, numeric, date, etc.) and, if

it is a string field, the maximum length of the data. Finally, sample field data is simply sample data that Seagate Crystal Reports can display in the preview window while you design the report.

For complete information on creating field definition files, see *Creating Field Definition Files*, Page 98. Seagate Crystal Reports installs a sample field definition file in the \CRW directory on your system. This file is named ORDERS.TTX and can be used with the Orders table in the CRAZE.MDB sample database.

Use the following examples as a guide for designing a field definition file:

Order ID	long	1
Customer Name	string 50	sample string value
Order Date	date	Jan 5, 1999
Order Amount	currency	1.00

The Active Data Driver supports the following data types in a field definition file:

<i>Data Type</i>	<i>Description</i>
Byte	8-bit integer value.
short, int16	16-bit integer value.
long, int32	32-bit integer value.
Number	64-bit floating-point value.
Date	Any date/time value. Examples include: <ul style="list-style-type: none">● Jan 5, 1999● 07/11/97 5:06:07● 07/11/97● 23:30:01
Currency	64-bit floating-point value that can include a currency or percent sign.
String	Any string value under 254 characters long, such as a name, description, or identification number that is not meant to be interpreted numerically. You must indicate the maximum number of characters for the string.
Bool	True/False Boolean value.
Memo	Any string value over 254 characters long. You must indicate the maximum number of characters for the string.
BLOB	Fields that contain bitmap images.

Design the Report

Using Seagate Crystal Reports, you can design the report files for your application based on the field definition file you created. Use the following steps as a guide for building a report file using the Active Data Driver:

1. Click New Report in the Seagate Crystal Reports Welcome dialog box, or click the New button on the Seagate Crystal Reports toolbar. The Report Gallery appears.
2. Click a Report Expert button in the Report Gallery. In this example, you can click Standard. The Create Report Expert appears.
3. On the Data Tab, scroll down until you see the Active Data button. If this button does not appear on the Data Tab, you have not correctly installed the Active Data Driver. Run Seagate Crystal Reports Setup again.
4. Click the Active Data button, and the Select Field Definition File dialog box appears.
5. Enter the path and file name of your field definition file, or use the Browse button to locate the file. You can also use the New button in this dialog box to create a new text file. For more information on using the New button in this dialog box, see *Database Definition Tool, Page 98*.
6. Click OK after you have selected your field definition file. The Choose SQL Table dialog box appears.
7. Make sure your field definition file is selected in the SQL Tables list box, and click Add. Click Done when finished. The name of your field definition file appears in the list of tables on the Data Tab of the Create Report Expert.
8. Click the Fields Tab. Field names defined in the field definition file appear in the Database Fields list box just as they do when you add a standard database table or ODBC data source to your report. Continue designing your report using the tools and options available in the Create Report Expert.
9. When you finish designing your report in the Create Report Expert, click Preview Report. The report appears in the Seagate Crystal Reports Preview Tab with the sample values you defined in the field definition file. Continue fine-tuning your report and save it for use in your application.

NOTE: Before saving your report, be sure to turn off the Save Data with Report option under the File menu. The sample data stored with the field definition file is unnecessary at runtime, and will only increase the size of your report file.

Obtain a Recordset from the Runtime Data Source

Once you have created the field definition file and designed a report based on that file, you can begin programming your application to obtain a recordset from an active data source, open the report file, set the report file to use the recordset object, then print or export the report file. This process requires using the functionality of the Crystal Active Data Driver in conjunction with the Crystal Report Engine.

Either the Crystal Report Engine API or the Crystal Report Engine Automation Server can be used with the Active Data Driver. However, the following tutorials use the Crystal Report Engine Automation Server in Visual Basic 4.0. This section assumes a familiarity with the Crystal Report Engine Automation Server. If you need more information on how to use the automation server, see *Crystal Report Engine Automation Server, Page 85*.

To begin, you must obtain a Recordset object from a runtime active data source. This data source can be opened through DAO, RDO, ADO, the Visual Basic Data Control, or Crystal Data Objects (CDO). For information on DAO, RDO, and ADO, refer to Microsoft documentation. For information on the Visual Basic Data Control, refer to your Visual Basic documentation. For information on CDO, see *Crystal Data Object, Page 100*.

This tutorial creates a Recordset object from the Orders table of the CRAZE.MDB sample database using DAO. If you are using RDO, you will need to obtain a rdoResultSet object. If you are using CDO, you will need to obtain a Rowset object (see *Crystal Data Object, Page 100*).

NOTE: You must add the Data Access Objects component to your Visual Basic project before performing the following steps. For instructions on using DAO with Visual Basic, refer to your Visual Basic documentation.

1. Declare variables for the Database and Recordset objects in your Visual Basic application. This can be handled in the declarations section of a form or module. Use code similar to this:

```
Dim db As Database  
Dim daoSet As Recordset
```

2. Obtain a Database object from the Craze database.

```
Set db = DBEngine.Workspaces(0).OpenDatabase("c:\crw\craze.mdb")
```

3. Obtain a Recordset object from the Orders table of the Craze database.

```
Set daoSet = db.OpenRecordset("Orders", dbOpenTable)
```

Open the Report

Once you have obtained a Recordset object, you can begin working with the report file you created earlier. This example uses the Crystal Report Engine Automation Server to open a report file.

NOTE: You must add the Crystal Report Engine Automation Server component to your Visual Basic project before performing the following steps. For complete information on using the Automation Server, see *Crystal Report Engine Automation Server, Page 85*.

1. Declare variables for the Application and Report objects that you will obtain from the Crystal Report Engine Object Library in the automation server. This can be handled in the declarations section of a form or module.

```
Dim app As Application  
Dim report As Report
```

2. Create an Application object with the Crystal Report Engine Automation Server.

```
Set app = CreateObject("Crystal.CRPE.Application")
```

3. Obtain a Report object by opening the ORDERS.RPT report file. This report file must be designed based on a field definition file (see *Creating Field Definition Files, Page 98*). The report you created in the section *Design the Report, Page 95*, will work for this step.

```
Set report = app.OpenReport("c:\reports\Orders.rpt")
```

Pass the Recordset to the Active Data Driver

The Recordset object gets passed to the Active Data Driver through the SetPrivateData method of the DatabaseTable object in the Crystal Report Engine Object Library. You must first obtain a DatabaseTable object from the Report object, then you must use the SetPrivateData method to set the report to point at the recordset object for your Active data source. The Crystal Report Engine Automation Server uses the Active Data Driver itself to replace the field definition file, at runtime, with the Active data source.

The following code demonstrates how to obtain a DatabaseTable object from the Report object:

```
Dim reportDb As Database
Dim reportTables As DatabaseTables
Dim reportTable As DatabaseTable

Set reportDb = report.Database
Set reportTables = reportDb.Tables
Set reportTable = reportTables.Item(1)
```

The Item property in the DatabaseTables collection lets you specify which table in the database you are replacing. Since the field definition file acts as a database with a single table, you should pass 1 to the Item property.

Once you have a DatabaseTable object for the Report object, you can pass the Active data source to the Report object using the SetPrivateData method. This method requires two parameters. The first is a value indicating that the data source you are passing to the report is an Active data source. This value must be 3. The second parameter is the data source itself. For example:

```
reportTable.SetPrivateData(3, daoSet)
```

Print the Report

Now that the data source for the report has been set to the DAO Recordset, you can print, preview, or export the report normally. For instance, the following code prints the report to the default printer:

```
report.PrintOut
```

Once the data source has been set in the report object, runtime reporting can proceed normally. All features of the Crystal Report Engine are available to you. For more

information, refer to the sections of this manual appropriate to the Report Engine development tool you are using.

Creating Field Definition Files

Although field definition files can be created manually using a text editor such as Notepad, Seagate Crystal Reports provides a few options for simplifying the process:

1. *Database Definition Tool, Page 98*
2. *Active Data Driver Functions, Page 99*
3. *Crystal ActiveX Control, Page 82*

Each tool has its advantages. Review the process for using each tool described below to determine which best suits your own environment and development process.

Database Definition Tool

The Database Definition Tool is available from the Create Report Expert when you begin designing your report. It allows you to design a field definition file as the first step of designing your report. From the Data Tab of the Create Report Expert:

1. Scroll down and click the Runtime DB button. The Select Field Definition File dialog box appears.
2. Click New to create a new field definition file. The Database Definition Tool appears.
3. Use the Database Definition Tool to create fields for your field definition file. Use the controls to enter a field name, field type, and sample data to appear in the Seagate Crystal Reports Preview Tab. If you select String as the field type, you will also be asked to specify a maximum string length.
4. Click Add to add the new field to your field definition file. The new field will appear in the list box at the bottom of the Database Definition Tool.
5. Continue adding as many fields as necessary for your field definition file by entering the information in the controls of the Database Definition Tool, and clicking Add each time.
6. You can delete a field that you have created by selecting the field in the list box and clicking Delete.
7. Click Done when you are finished designing your field definition file using the Database Definition Tool. A File Save dialog box appears.
8. Save the field definition file where it can be accessed by your report file. When finished, the new field definition file will appear in the Select Field Definition File dialog box.
9. Click OK in the Select Field Definition File dialog box, and continue designing your report.

Active Data Driver Functions

The Active Data Driver (P2SMON.DLL) is a standard dynamic link library that is normally used by Seagate Crystal Reports (or the Crystal Report Engine) to access active data sources such as DAO and ADO Recordsets. The DLL is installed, by default, in your \WINDOWS\SYSTEM directory. In addition, the Active Data Driver exports functions that can be used at runtime from within your application to dynamically design a field definition file based on your data source, and a report file based on the field definition file. These functions are available to any development environment that supports DLL function calls.

NOTE: To use the functions in the Active Data Driver DLL, you must declare the functions first. Refer to your Visual Basic documentation for information on declaring DLL functions. Search for Crystal Active Data Driver Reference in Developer's online Help for information about declaring the Active Data Driver functions.

To use the Active Data Driver Functions from Visual Basic:

1. Obtain a valid Recordset object from your DAO, ADO, or Data Control data source, or a valid Rowset object using CDO.
2. Call the function CreateReportOnRuntimeDS to create a field definition file base on your Recordset or Rowset object. For example:

```
CreateReportOnRuntimeDS(DAOSet, "c:\reports\orders.rpt",
                      "c:\reports\orders.ttx", True, False)
```

This example creates a field definition file named ORDERS.TTX, then creates a simple report file based on this field definition file and names it ORDERS.RPT. If the last argument is set to True, Seagate Crystal Reports, if installed, will open automatically on the user's machine, allowing them to make changes to the report file.

Notice that the first argument is a DAO Recordset object. If you are using this function in a language such as C or C++, you would pass a pointer to an IUnknown derived interface to the Recordset.

NOTE: For complete information on the functions provided by the Active Data Driver, search for Crystal Active Data Driver Reference in Developer's online Help.

Visual InterDev Design-time ActiveX Control

The Visual InterDev Design-time ActiveX Control provides a graphical user interface at design time through which you can create a report based on a field definition file that is applied to an Active data source at runtime. The Design-time Control is designed primarily for developers building web sites using Microsoft Visual InterDev. For complete information on building Active web sites using Visual InterDev, the Active Data Driver, and the Visual InterDev Design-time ActiveX Control, see *Building Active Web Sites, Page 25*.

Crystal Data Object

The Crystal Data Object (CDO) is a new ActiveX data source that allows you to define fields and records at runtime based on data that exists only at runtime. Through CDO, any data can become a virtual database and can be reported on using the power of the Crystal Report Engine.

CDO, like DAO and ADO, is based on the Component Object Model (COM). Any development environment that supports COM interfaces can dynamically generate a set of data for a report without relying on a database that exists at design time.

Applications that produce data that does not exist outside of the running application have been unable, until now, to take advantage of the most powerful reporting features in the industry. CDO, however, solves that problem. For instance, applications that monitor system or network resources, or any constantly operating environment, can produce a current report on such information at any time. No longer does data need to be dumped to a separate database before analysis. Through CDO, the Active Data Driver, and the Crystal Report Engine, analysis is instant and up-to-date.

Using the Crystal Data Object

The Crystal Data Object is an automation server that can be accessed from any Windows development environment that supports ActiveX. By creating a Rowset object, similar to a Recordset, and filling it with fields and data, you design a virtual database table that can be passed as an active data source to the Crystal Active Data Driver.

Once the CDO Rowset has been created, it can be used just like any other active data source such as DAO or ADO. Use a procedure, much like the procedure described in *Using the Active Data Driver, Page 93*, to print, preview, or export a report at runtime that is based on the CDO data source. Simply replace the steps that explain how to pass a DAO Recordset to the Active Data Driver with appropriate steps for passing your CDO Rowset.

The rest of this section explains how to create a CDO Rowset in Visual Basic 4.0. However, as an ActiveX automation server, CDO can be used by any application development environment that supports ActiveX.

To create a CDO Rowset:

1. *Obtain a CDO Rowset Object, Page 100*
2. *Add Fields to the Rowset Object, Page 101*
3. *Obtain Data as Rows, Page 101*
4. *Add Rows to the Rowset Object, Page 102*

Use these steps as a guideline for creating your own CDO Rowsets for use with the Active Data Driver.

Obtain a CDO Rowset Object

As stated earlier, CDO is a standard automation server. A Rowset object can be obtained from CDO using the Visual Basic CreateObject function:

```
Public CDOSet As Object  
Set CDOSet = CreateObject("Crystal.CDO.Rowset")
```

This Rowset object is, essentially, equivalent to a Recordset object you might obtain from DAO or another active data source. It is the Rowset object that you eventually pass to the Active Data Driver.

Add Fields to the Rowset Object

Once you have a Rowset object, you need to define fields for the Rowset. These fields act as the virtual database fields. The field names you specify must match the field names specified in the field definition file. For more information on field definition files, see *Creating Field Definition Files, Page 98*.

Fields are added to a CDO Rowset using the AddField method:

```
CDOSet.AddField "Order ID", vbString  
CDOSet.AddField "Company Name", vbString  
CDOSet.AddField "Order Date", vbDate  
CDOSet.AddField "Order Amount", vbCurrency
```

This code adds four fields to the Rowset with the specified field names, and field types. The field types are based on constant values for the Variant data type. The constant names used here are from Visual Basic. For information on valid constant values, see the AddField method in the *Crystal Data Object Reference* in Developer's online Help.

Obtain Data as Rows

Data to be added as rows in the Rowset can be collected in a two dimensional array. The first dimension indicates rows, while the second dimension specifies fields for each row. The number of possible fields indicated by the second dimension must not exceed the number of fields you added to the Rowset using the AddField method. For example, you might define an array such as this:

```
Dim Rows(12, 4) As Variant
```

This specifies an array named Rows that contains 12 rows (0 to 11) and 4 columns (0 to 3). Notice that the four fields are defined with the AddField method, so the 4 columns in the Rows array are also defined. In addition, room has been made for 12 rows or records. Finally, since each field holds a different type of data, the array is defined as a Variant type.

NOTE: If your Rowset contains only a single field, you can use a one dimensional array instead of two dimensional. The single dimension indicates the number of rows or records in your Rowset.

Now that you have defined an array to hold data, you can begin adding values to the array. These array values will become the actual field values for the virtual database. Most likely, you will want to design a routine in your application that adds runtime data generated by your application into each cell of the array. The following code, however, demonstrates how you can explicitly add values to the array:

```
Rows(0, 0) = "1002" 'The first Order ID  
Rows(0, 1) = "Cyclist's Trail Co." 'The first Company Name  
Rows(0, 2) = #12/2/94# 'The first Order Date  
Rows(0, 3) = 5060.2725 'The first Order Amount
```

From here, you could continue by adding a value to the first field of the second record, Rows(1, 0). You continue filling in data record by record and field by field. This technique, of course, requires a lot of code and is not very practical. Most real applications would contain a looping procedure that progressively filled in values for the array.

Add Rows to the Rowset Object

At this point, you have created a CDO Rowset object, added fields to the Rowset, and collected data in an array that will become part of a virtual runtime database. All that is left is to pass the data from the array to the Rowset object. This step is handled with a single method:

```
CDOSet.AddRows Rows
```

The AddRows method accepts a two dimensional array containing the values you want added to the Rowset and, ultimately, added to a report file that is printed or exported.

Rows can be added to a CDO Rowset with multiple calls to the AddRows method. However, once you begin adding rows of data to a Rowset, you can not add any new fields to the Rowset. Any call to AddFields after a successful call to AddRows will fail.

Once you finish populating your virtual database in the CDO Rowset object, you can pass this object as an active data source to the Active Data Driver using the SetPrivateData method in the Crystal Report Engine Automation Server. For complete instructions on doing this, see *Pass the Recordset to the Active Data Driver, Page 97*.

Crystal Data Object Library

The Rowset object and its related objects, methods, and properties are documented in the Seagate Crystal Reports Developer's Online Help. Search for the *Crystal Data Object Reference* in the Help file for a complete description of the entire Crystal Data Object Library.

Using Grid Controls with the Crystal Report Engine

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

In Seagate Crystal Reports, a Crystal ActiveX Control can be bound directly to a Visual Basic Data Control. Using the Visual Basic Data Control with the Crystal ActiveX Control offers the following benefits:

- Generating reports in Visual Basic programs is made even easier and does not require an existing .RPT file.

- A powerful feature of Visual Basic is ad-hoc queries that are run by executing SQL statements in the RecordSource property of the Data Control. By directly binding a Crystal Custom Control to a Data Control, users can now create reports of dynaset data which are the results of such ad-hoc queries.

Bound Report Driver and Bound Report Files

When using Seagate Crystal Reports to generate reports from database files of a particular file format (i.e., Paradox file format), you need to have the appropriate report driver (i.e., PDBPDX.DLL) to retrieve data from the databases. Similarly, when you generate reports by binding to a Visual Basic Data Control, a Bound Report Driver (PDBBND.DLL) is used to retrieve data from the Data Control. Make sure PDBBND.DLL is in your \WINDOWS\SYSTEM directory or search paths, along with other database drivers.

Crystal ActiveX Control Properties

Several properties are added to the Crystal Custom Control in order to support bound reports. These new properties are described below.

Custom

This property allows you to create bound .RPT files at Visual Basic design time and is not available at runtime. After a bound .RPT file is created, programmers can then use Seagate Crystal Reports to customize the report layout or even link the bound data to other database tables.

DataSource (Data Control)

This property can be read/write at design time and runtime. This property is ignored if the ReportSource property is 0 (Report files). To generate bound reports, set this property to the Data Control you want to retrieve data from. The Data Control must already be on the form before this property can be set.

BoundReportFooter (Boolean)

This property can be read/write both at design-time and runtime. This property is ignored if the ReportSource property is 0 (Report files). Default is *False* and the bound reports generated will not have page numbers printed. If set to *True*, page numbers will be printed at the bottom of a page.

BoundReportHeading (string expression)

This property can be read/write both at design time and runtime. This property is ignored if the ReportSource property is 0 (Report files). It specifies the report title at the beginning of a bound report. If it is blank, no report title will be printed.

ReportSource (numeric expression)

This property can be read/write both at design time and runtime. The allowed values are:

- 0 - Report files**
- 1 - Bound TrueDBGrid Control**
- 3 - All Data Control Fields**

The default value is 0 - Report files, and the ReportFileName property must be assigned to an existing report path name (.RPT). This is equivalent to when the new bound report features were not available and all reports were generated from existing .RPT files.

When set to 1 or 3, the ReportFileName property will be ignored and no .RPT file is needed. Reports will be created using data retrieved from Data Control. The reports generated directly from the Data Control are identical to the reports generated from the respective bound .RPT files created using the (Custom) property described above.

Creating a Bound Report using the Crystal ActiveX Control

1 Add the following controls to your Visual Basic form:

- Data Control
- Crystal ActiveX Control
- Command Button

2 On the Data Control:

- Set the DatabaseName property to the name of the database being reported on.
- Set the RecordSource property (this can be a database table or a SQL query statement).

3 On the Crystal ActiveX Control:

- Set the DataSource property to the Data Control (for example, Data1).
- Set the ReportSource to 3 - All Data Control Fields.

4 On the Command Button, add the following code for the Click event:

```
Private Sub Command1_Click()
    CrystalReport1.Action = 1
End Sub
```

Run the application, click the command button, and the Crystal ActiveX Control will retrieve data from the Data Control and create the report. The report will appear as a simple columnar report. There are no runtime properties to control any report formatting. However, this can be accomplished at design-time by editing the report designed by the ActiveX control (a report template) in Seagate Crystal Reports.

Creating a Formatted Bound Report

- 1 Add the Data control, ActiveX control, and a command button to your form.
- 2 On the Data control, set the DatabaseName property and the RecordSource property as you did in the previous example.
- 3 On the ActiveX control:
 - Set the DataSource property to the Data Control (i.e., Data1).
 - Set the ReportSource property to 3 - All Data Control Fields.
 - Open the Custom property and select the Data-Bound Report Tab.
 - Click the Save Report As button and enter a name for the report.
- 4 Open the report template in Seagate Crystal Reports and apply any formatting that you want including spacing between columns, font size, colors, grouping, and totaling. Save the report template again when finished.
- 5 In your Visual Basic application, set the following properties for the ActiveX control:
 - Set the ReportSource to 0 - Report File.
 - Set the ReportFileName to the .RPT file that you saved (include the complete path of the file).
- 6 On the command button, add the following code to the Click event:

```
Private Sub Command1_Click()  
    CrystalReport1.Action = 1  
End Sub
```

Now, the application will create the report at runtime with the formatting you have applied.

Creating a Formatted Bound Report at Runtime

The following steps describe an alternative method of creating formatted bound reports:

- 1 Create your Visual Basic application as in the first example above.
- 2 Set the ActiveX Control to print to a preview window, and run the application.
- 3 Click the Export button in the preview window, and export the report to a disk file in .RPT format.
- 4 Once the report has been exported, you can open it up in Seagate Crystal Reports.
- 5 Perform all formatting changes that you want and save the report.
- 6 Return to the Visual Basic application and stop it if it is still running.
- 7 On the ActiveX Control:
 - Set the ReportSource to 0 - Report File.
 - Set the ReportFileName to the .RPT file that you created.

- 8 Run the Visual Basic application and you will be able to see your bound report with the formatting changes you've made.

NOTE: When creating formatted reports for use with the bound data control in Visual Basic, you will not be able to refresh the data from within Seagate Crystal Reports since the data does not exist outside of the Visual Basic application.

NOTE: If you plan on using a formatted bound report, you will not be able to modify anything in the SELECT statement of the data control. The report needs all these fields and will fail if they are not all there. The formatted report can not report on any new fields.

When passing properties at runtime using bound reports (i.e., SortFields), the syntax is slightly different. For example, the following syntax would be used for the Formulas and SortFields properties in a normal report:

```
CrystalReport1.Formulas(0) = "COMMISSION= {TableName.FIELDNAME}"  
CrystalReport1.SortFields(0) = "+{TableName.FIELDNAME}"
```

However, for a bound report, the following syntax would be used:

```
CrystalReport1.Formulas(0) = "COMMISSION= {Bound Control.FIELDNAME}"  
CrystalReport1.SortFields(0) = "+{Bound Control.FIELDNAME}"
```

Sample Application

Seagate Crystal Reports includes a complete sample application written in Visual Basic 5.0 using the Crystal Report Engine Automation Server and the Microsoft Data Bound Grid control. The Xtreme Mountain Bike Inventory Application is a complete real-world application that provides various reports to employees at a fictitious company. The Microsoft Data Bound Grid control is used for an order-entry page that dynamic reports are produced from. The application is installed, by default, in the \CRW\SAMPLE\XTREME\INVNTORY directory.

6 Solutions for Other Development Tools

What you will find in this chapter...

Using the Crystal Report Engine API in Delphi, Page 108

The Crystal Visual Component Library, Page 108

The Crystal Report Engine Class Library, Page 110

The Crystal NewEra Class Library, Page 112

Using the Crystal Report Engine API in Delphi

All versions of Delphi can make direct calls to the functions in the Crystal Report Engine API. The Delphi unit file CRPE.PAS (CRPE32.PAS for 32-bit systems) includes complete declarations for all Report Engine API functions and records. When you need to add the Report Engine API to your own Delphi unit, simply add the Crystal Report Engine API unit to your project and refer to the unit in your *uses* clause. For example:

```
Uses  
  crpe32;
```

The *implementation* section of the Crystal Report Engine API unit contains all of the Crystal Report Engine API functions defined as *external* and as part of the CRPE or CRPE32 DLL.

The *Crystal Report Engine Reference, Page 115*, includes Delphi declarations for all Report Engine API functions and records. In addition, the Developer's online Help includes Delphi sample code using many of the functions and records defined in CRPE.PAS. Search for *Report Engine Functions - Sample Code in Delphi* in the Developer's online Help.

NOTE: Many functions and records are defined differently for 16-bit and 32-bit systems. When referring to declarations in reference sections, make sure you are using the correct version of the function or record for the operating system environment you are working in.

The Crystal Visual Component Library

The Crystal Visual Component Library (VCL) has been designed specially for the Borland Delphi development environment. The VCL can be easily added to any Delphi project and compiled into your final executable application. Like the ActiveX control, the VCL provides all of the report processing power available in Seagate Crystal Report for your Delphi projects.

NOTE: Delphi 2.0 and later can also use ActiveX controls. Refer to your Delphi 2.0 documentation on how to use an OLE Control (OCX) in a Delphi project.

Adding the VCL to your Project

The following instructions demonstrate how to add the Crystal VCL to a Delphi 2.0 project using the Install command on the Component menu. If you are using Delphi 1.x, use the Install Components command on the Options menu and refer to your Delphi 1.0 documentation for complete instructions.

- 1 With your project open in Delphi, choose the Install command from the Component menu. The Install Components dialog box appears.
- 2 Click Add to add a new component to your project. The Add Module dialog box appears.
- 3 Click Browse. The Add Module dialog box changes to allow you to browse through your directories and files to find a component module.

- 4 Select Unit file (*.DCU) from the Files of type drop-down box. The Crystal VCL is a Delphi Unit file.
- 5 Use the controls in the Add Module dialog box to locate and select the Crystal VCL. The Crystal VCL was installed in the Seagate Crystal Report VCL directory (\CRW\VCL by default) when you installed Seagate Crystal Reports. The file is named UCRPE.DCU (16-bit) or UCRPE32.DCU (32-bit).
- 6 Once you select the VCL file, click Open. The Add Module dialog box closes.
- 7 In the Install Components dialog box, the name of the Crystal VCL (UCRPE or UCRPE32) will appear in the Installed units list box. Select this VCL, and the TCrpe class will appear in the Component classes list box. Click OK to close the Install Components dialog box when finished. Delphi will recompile the component library, adding in the Crystal VCL.
- 8 Click the Data Access Tab on the Component Palette. The Crystal VCL appears as the last component on the Data Access Tab. When you hold the cursor over the component, a Tool Tip appears indicating the name of the component as Crpe.
- 9 Double-click the Crpe component to add it to the active form in your project.

Using the Crystal VCL

Once you add the Crpe component to a form in your project, you build the connection between your Delphi application and the Crystal Report Engine by setting the component's properties via the Object Inspector. Using the Object Inspector, you specify:

- the name of the report you want to print in response to an application event,
- the destination for that report (window, file, or printer),
- the number of copies you want to print (if your report is going to the printer),
- print file information (if your report is going to a file),
- preview window sizing and positioning information (if your report is going to a window),
- selection formula information (if you want to limit the records in your report),
- sorting information, and
- other related properties.

Crpe component properties can be changed either at design time or at runtime. Note, however, some properties are available only at runtime. These properties will not appear on the Properties list in the Object Inspector.

NOTE: For a complete description of each property in the Crystal VCL, refer to Visual Component Library Reference, Page 739.

Changing Properties in the Object Inspector

To change the value for a property, click the property and then do the following:

- If a text box appears next to the property name, type in a value for the property.
- If a drop down list box appears next to the property name, click the arrow to reveal your alternatives and select a value from the list.
- If a text box with an ellipsis (...) button appears next to the property name, click the button to reveal a dialog box you can use to define your setting for the property.

Changing Properties at Runtime

You can set most of the properties for the Crpe component at runtime by adding simple entries to your procedure code. Runtime property settings replace settings you make via the Object Inspector at design time.

The Execute property is used to actually process the report at runtime. This property can only be set at runtime, and it is the only means by which a report can actually be generated by the Crpe component.

For information on how to set component properties at runtime, refer to your Delphi documentation. Developer's online Help contains code examples for each of the Crpe component properties. Search for the property you are interested in for examples.

The Crystal Report Engine Class Library

The Crystal Report Engine Class (REC) Library provides object oriented programming of the Crystal Report Engine in C++. The Crystal REC Library is based on the Microsoft Foundation Class (MFC) Library. The two Crystal Report Engine Class definitions in the Crystal REC Library are derived from the MFC CObject class.

MFC is a widely available, and highly powerful class library originally designed for use with Microsoft's C++ development environment (Microsoft Visual C++). However, MFC is a complete class library designed for programming Windows applications and can be used with most C++ development environments.

The Crystal REC Library is comprised of two primary classes and several supporting structures. These classes are wrapped around the Crystal Report Engine API, providing an object oriented approach to programming the Crystal Report Engine.

NOTE: The Crystal REC Library can be used with any available version of the MFC Library.

The Crystal REC Library was installed in the same directory as Seagate Crystal Report (\CRW by default). You need to add two files to your C++ project to use the Crystal REC Library:

PEPLUS.H

This C++ header file contains the Crystal Report Engine class definitions along with several structure definitions used by the Crystal Report Engine classes. You must #include this file in any code module that will be using the Crystal Report Engine classes.

PEPLUS.CPP

This C++ source code file contains implementations of all of the class methods for the Crystal REC Library classes. This file must be added to your C++ project file in order to use the Crystal REC Library. This code is compiled directly into your Visual C++ project and calls the functions available in the Crystal Report Engine.

The Crystal Report Engine API

As mentioned above, the Crystal REC Library classes are wrapped around the Crystal Report Engine API, and the methods in these classes make direct calls to the functions in the Crystal Report Engine API. For this reason, your project must include the CRPE32M.LIB Crystal Report Engine library. Make sure CRPE32M.LIB has been added to the list of libraries included with your C++ project.

NOTE: If you are programming for a 16-bit platform (Windows 3.1x), use the 16-bit version of the Crystal Report Engine library, CRPE.LIB.

NOTE: For complete descriptions of each of the REC classes, their respective members, and all supporting structures, refer to The Crystal Report Engine Class Library Reference, Page 839.

Classes in the Crystal Report Engine Class Library

The Crystal REC Library consists of two primary classes. The CRPEngine class controls the entire Crystal Report Engine. It is designed so that there should only be one CRPEngine object in the entire application. The CRPEngine object contains methods that are common to all print jobs (i.e., SQL connections, version information, etc.). More importantly, it is responsible for creating and managing all CRPEJob objects.

The CRPEJob class controls print jobs. A print job is a request for a report to be processed and printed, previewed, or exported. You do not construct a CRPEJob object directly; instead, you request a job instance from the CRPEngine class and receive a pointer to a CRPEJob object. It is the CRPEJob object that allows you access to the attributes of a print job.

Both classes are derived from the MFC CObject class and provide all of the functionality of that class, including runtime class information and object diagnostic output.

The Crystal NewEra Class Library

The Crystal Class Library for NewEra is designed specifically for use with the Informix NewEra development environment. This class library is comprised of two primary classes and several supporting classes. These classes are wrapped around the Crystal Report Engine API and insulate the user from some of the details required to manage reports in an application.

The Crystal Class Library for NewEra was installed in the Crystal Informix directory (\CRW\INFORMIX by default) when you installed Seagate Crystal Reports. The header files (*.4GH) for the library are located in \CRW\INFORMIX\CLASS\INCL, while source files (*.4GL) are located in \CRW\INFORMIX\CLASS\SRC. Copy these files, along with CRPE.H, to the appropriate location for your NewEra application, then include CRYSTAL.4GH in the source file for your application project that will access the Crystal Report Engine.

Crystal Report Engine API

As mentioned above, the classes in the Crystal Class Library for NewEra are wrapped around the Crystal Report Engine API, and the methods in these classes make direct calls to the functions in the Crystal Report Engine API.

Classes in the Crystal Class Library for NewEra

The Crystal Class Library for NewEra consists of two primary classes. The CRPEngine class is designed so that there should only be one CRPEngine object in the entire application. The CRPEngine object contains methods that are common to all print jobs (e.g., SQL connections, version information, etc.). More importantly, it is responsible for creating and managing all CRPEJob objects. It is the CRPEJob object that allows you access to the attributes of a print job.

Part 3 - Crystal Report Engine Reference

7

Crystal Report Engine Reference

What you will find in this chapter...

Working with section codes, Page 116

FUNCTIONS, Page 121

STRUCTURES, Page 353

Working with section codes

A report, by default, contains five areas: Report Header, Page Header, Details, Report Footer, and Page Footer. Each of those areas can contain one or more sections. When you add groups, subtotals, or other summaries to your report, the program adds Group Header and Group Footer areas as needed, and each of those areas can contain one or more sections as well. Since one report can have a totally different section configuration than the next, Seagate Crystal Reports uses calculated section codes to identify the sections in each report.

In Crystal Report Engine API functions that affect report sections, the *sectionCode* parameter encodes the section type, the group number (if the section is a Group Header or Group Footer section), and the section number (if there are multiple sections in an area) together in a single value.

The Crystal Report Engine API also includes macros for encoding section codes (PE_SECTION_CODE, for use with functions that require a section code) and for decoding section codes (PE_SECTION_TYPE, PE_GROUP_N, and PE_SECTION_N, for use with functions that return a section code). The examples that follow show how the encoding and decoding macros can be used.

NOTE: You can not pass the above values directly to a function as section codes. You must use the encoding macro to create a valid section code based on one of the above constants.

Encoding

The PE_SECTION_CODE macro allows you to define a section code to pass as a parameter in Crystal Report Engine functions that require a section code. The syntax for the macro is:

```
PE_SECTION_CODE (sectionType, groupNumber, sectionNumber)
```

sectionType

This indicates the report area or section type that the section is in. For section type, use any of the following constants

Section Type Constant	Value	Description
PE_SECT_REPORT_HEADER	1	Report Header Section
PE_SECT_PAGE_HEADER	2	Page Header Section
PE_SECT_GROUP_HEADER	3	Group Header Section
PE_SECT_DETAIL	4	Detail Section
PE_SECT_GROUP_FOOTER	5	Group Footer Section
PE_SECT_PAGE_FOOTER	7	Page Footer Section
PE_SECT_REPORT_FOOTER	8	Report Footer Section
PE_ALLSECTIONS	0	All Report Sections

groupNumber

Indicates which group the section is in. If the sectionType value indicated is PE_SECT_GROUP_HEADER or PE_SECT_GROUP_FOOTER, the groupNumber is a zero (0) based index for the group section. If the sectionType value is not one of these group section constants, the groupNumber value should always be zero.

sectionNumber

If the report area has been split into more than one section, sectionNumber indicates which section within the area you are using. This value is a zero (0) based index. In other words, the first section in an area is 0, the next section is 1, etc.

NOTE: *The macro PE_SECTION_CODE calculates and returns the section code number; it does not return an error code.*

The following example demonstrates how to obtain a section code using the PE_SECTION_CODE macro. The section code obtained here is for the second section in the Group Header 1 area:

```
code = PE_SECTION_CODE(PE_SECT_GROUP_HEADER, 0, 1);
PESetSectionFormat(job, code, &mySectionOptions);
```

In this case you pass the section type (PE_SECT_GROUP_HEADER), the group number (since this is the first group, use the zero indexed group number 0) and section number (since this is the second section in the Group Header, use the zero indexed section number 1). The program uses the encoding macro and returns a section code which is then passed in the PESetSectionFormat call.

When using PE_ALLSECTIONS in your macro, code can be written in one of two ways:

```
code = PE_SECTION_CODE(PE_ALLSECTIONS, 0, 0);
// the code value returned is 0 - NOT an error code
PESetSectionFormat(job, code, &mySectionOptions);
```

or, you can eliminate using the macro all together:

```
PESetSectionFormat(job, PE_ALLSECTIONS, & mySectionOptions)
```

NOTE: *The maximum number of groups is 25 (possible values of 0 to 24). The maximum number of sections is 40 (possible values of 0 to 39).*

Decoding

Some Crystal Report Engine functions return section codes. These values can be decoded using one of three macros:

1. PE_SECTION_TYPE (sectionCode)
2. PE_GROUP_N (sectionCode)
3. PE_SECTION_N (sectionCode)

Each macro accepts an encoded section code as a parameter.

In the following example, you determine the number of sections in the report (using PEGetNSections), obtain the section code for each section (using PEGetSectionCode), and then decode the section code using the PE_SECTION_TYPE, PE_GROUP_N, and PE_SECTION_N macros.

```
numSections = PEGetNSections(job);
for (i = 0;i < numSections;i++)
{
    code = PEGetSectionCode(job, loopSectionN);
    areaType = PE_SECTION_TYPE(code);
    groupN = PE_GROUP_N(code);
    sectionN = PE_SECTION_N(code);

    // Perform section specific code here
}
```

Once you've identified the area, group, and section you want, you can then set the section format using code similar to this:

```
PESetSectionFormat(job, code, &mySectionOptions);
```

NOTE: Earlier versions of Seagate Crystal Reports used different section code constants. Those constants have been remapped to the new section code format so reports created with earlier versions of Seagate Crystal Reports can run with applications created with the current version.

Section Map

The following map shows the pattern of section code assignment:

<i>Report Header</i>	
1000	First Section in Report Header Area
1025	Second Section in Report Header Area
1050	Third Section in Report Header Area
1075	Fourth Section in Report Header Area
up to 1975	40th Section in Report Header Area
<i>Page Header</i>	
2000	First Section in Page Header Area
2025	Second Section in Page Header Area
2050	Third Section in Page Header Area
2075	Fourth Section in Page Header Area
up to 2975	40th Section in Page Header Area
<i>GH1</i>	
3000	First Section in First Group Header Area
3025	Second Section in First Group Header Area
3050	Third Section in First Group Header Area
3075	Fourth Section in First Group Header Area
<i>GH2</i>	
3001	First Section in Second Group Header Area
3026	Second Section in Second Group Header Area
3051	Third Section in Second Group Header Area
3076	Fourth Section in Second Group Header Area
<i>Details</i>	
4000	First Section in Details Area
4025	Second Section in Details Area
4050	Third Section in Details Area
4075	Fourth Section in Details Area
<i>GF1</i>	
5000	First Section in First Group Footer Area
5025	Second Section in First Group Footer Area

5050	Third Section in First Group Footer Area
5075	Fourth Section in First Group Footer Area
<i>GF2</i>	
5001	First Section in Second Group Footer Area
5026	Second Section in Second Group Footer Area
5051	Third Section in Second Group Footer Area
5076	(Fourth Section in Second Group Footer Area
<i>Page Footer</i>	
7000	First Section in Page Footer Area
7025	Second Section in Page Footer Area
7050	Third Section in Page Footer Area
7075	Fourth Section in Page Footer Area
<i>Report Footer</i>	
8000	First Section in Report Footer Area
8025	Second Section in Report Footer Area
8050	(Third Section in Report Footer Area
8075	Fourth Section in Report Footer Area

FUNCTIONS

PECancelPrintJob

Cancels the printing of a report. This function can be tied to a control that allows the user to cancel a print job in progress.

Use PECancelPrintJob to replace the Cancel button when PEShowPrintControls disables the Print Control buttons.

```
void CRPE_API PECancelPrintJob (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job.

Return Value

None

PECancelPrintJob (Other Declares)

VB Syntax

16-bit

```
Declare Sub PECancelPrintJob Lib "crpe.dll" (ByVal printJob As Integer)
```

32-bit

```
Declare Sub PECancelPrintJob Lib "crpe32.dll" (ByVal printJob As Integer)
```

Delphi Syntax

16-bit

```
procedure PECancelPrintJob (
    printJob: integer
);
```

32-bit

```
procedure PECancelPrintJob (
    printJob: Word
)stdcall;
```

dBASE for Windows Syntax

```
EXTERN CVOID PEcancelPrintJob (CWORD) CRPE.DLL
```

PECanCloseEngine

Determines whether or not the Crystal Report Engine can be closed. Use this function before calling *PECloseEngine*, *Page 127*, to verify that the engine is no longer processing print jobs. If the Crystal Report Engine closes while a print job is still running, an error can occur in your application or on the user's system.

```
BOOL CRPE_API PECanCloseEngine (void);
```

Return Value

TRUE (1) if the Engine can be closed; FALSE (0) if the Engine is busy.

PECanCloseEngine (Other Declares)

VB Syntax

16-bit

```
Declare Function PECanCloseEngine Lib "crpe.dll" () As Integer
```

32-bit

```
Declare Function PECanCloseEngine Lib "crpe32.dll" () As Integer
```

Delphi Syntax

16-bit

```
function PECanCloseEngine:  
    Bool;
```

32-bit

```
function PECanCloseEngine:  
    Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PECanCloseEngine () CRPE.DLL
```

PECheckFormula

Checks the text of a named formula for validity. Use this function to check a named formula for errors. This function works like the Check button in the Formula Editor. If the named formula contains an error, the function returns False.

```
BOOL CRPE_API PECheckFormula (  
    short printJob,  
    char FAR *formulaName  
) ;
```

Parameters

Parameters	Description
printJob	The handle of the print job containing the named formula you are checking.
formulaName	The name of the formula you wish to check for errors.

Return Value

TRUE (1) if the formula is correct; FALSE (0) if the formula has an error.

Remarks

- When specifying the name of the formula, do not use the @ symbol before the name. The @ symbol is only used by Seagate Crystal Reports to separate a formula field from other types of fields.

- PECheckFormula works like the Check button that appears in the Seagate Crystal Reports Formula Editor.

PECheckFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PECheckFormula Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal FormulaName As String) As Integer
```

32-bit

```
Declare Function PECheckFormula Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal FormulaName As String) As Integer
```

Delphi Syntax

16-bit

```
function PECheckFormula (  
    printJob: integer;  
    formulaName: PChar  
) : Bool;
```

32-bit

```
function PECheckFormula (  
    printJob: Word;  
    formulaName: PChar  
) : Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PECheckFormula (CWORD, CSTRING) CRPE.DLL
```

PECheckGroupSelectionFormula

Checks the text of the report's group selection formula for errors. Use this function when the group selection formula in a report has changed and you need to check the new group selection formula. If there is an error in the formula, this function returns a False value.

```

BOOL CRPE_API PECheckGroupSelectionFormula (
    short printJob
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	The handle of the print job for the report containing the group selection formula you wish to check.

Return Value

TRUE (1) if there are no errors in the group selection formula; FALSE (0) if the group selection formula contains an error.

Remarks

This function works just like the Check button that appears in the Seagate Crystal Reports Formula Editor.

PECheckGroupSelectionFormula (Other Declares)

VB Syntax

16-bit

```

Declare Function PECheckGroupSelectionFormula Lib "crpe.dll"
    (ByVal printJob As Integer) As Integer

```

32-bit

```

Declare Function PECheckGroupSelectionFormula Lib "crpe32.dll"
    (ByVal printJob As Integer) As Integer

```

Delphi Syntax

16-bit

```

function PECheckGroupSelectionFormula (
    printJob: integer
): Boolean;

```

32-bit

```
function PECheckGroupSelectionFormula (
    printJob: Word
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PECheckGroupSelectionFormula (CWORD) CRPE.DLL
```

PECheckSelectionFormula

Checks the text of the report's record selection formula for errors. Use this function whenever the record selection formula has been changed and you wish to check the formula for syntax errors. If the record selection formula contains an error, this function returns a False value.

```
BOOL CRPE_API PECheckSelectionFormula (
    short printJob
);
```

Parameters

Parameter	Description
printJob	The handle of the print job containing the record selection formula that you are checking.

Return Value

TRUE (1) if the selection formula does not have an error; FALSE (0) if the selection formula contains an error.

Remarks

The PECheckSelectionFormula function works like the Check button that appears in the Seagate Crystal Reports Formula Editor.

PECheckSelectionFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PECheckSelectionFormula Lib "crpe.dll" (ByVal  
printJob As Integer) As Integer
```

32-bit

```
Declare Function PECheckSelectionFormula Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PECheckSelectionFormula (   
    printJob: integer  
 ): Bool;
```

32-bit

```
function PECheckSelectionFormula (   
    printJob: Word  
 ): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PECheckSelectionFormula (CWORD) CRPE.DLL
```

PECloseEngine

Terminates the Crystal Report Engine. All printing is stopped and all windows are closed.

NOTE: This will only happen if the calling application is the last one using CRPE.

This function stops the Crystal Report Engine from sending output, but the report may continue to print from data remaining in the spooler.

This function is a necessary part of any custom-print link. It is also required for any print-only link in which you want the report to print to a window that is to remain visible after the report is printed. It is not necessary to use this function with a print-only link when directing the report to a printer.

```
void CRPE_API PECloseEngine();
```

Return Value

None

Remarks

Once this function has been called, no other Crystal Report Engine functions can be called except *PFOpenEngine*, *Page 247*. Call *PECanCloseEngine*, *Page 122*, before calling this function to make sure no print jobs are in a busy state.

PECloseEngine (Other Declares)

VB Syntax

16-bit

```
Declare Sub PECloseEngine Lib "crpe.dll" ()
```

32-bit

```
Declare Sub PECloseEngine Lib "crpe32.dll" ()
```

Delphi Syntax

16-bit

```
procedure PECloseEngine;
```

32-bit

```
procedure PECloseEngine  
  stdcall;
```

dBASE for Windows Syntax

```
EXTERN CVOID PECloseEngine () CRPE.DLL
```

PEClosePrintJob

Closes the print job. If printing has not yet finished, it continues; if the preview window is open, it stays open. This function is used as a mandatory part of each Custom-Print link to shut down the print job once it has finished printing to screen or to window.

```
BOOL CRPE_API PEClosePrintJob (
    short printJob
);
```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job you want to close.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

Once this function has been called, most other Crystal Report Engine functions, except *PEOpenPrintJob*, *Page 248*, *PECloseEngine*, *Page 127*, *PEGetVersion*, *Page 231*, and *PELogOnServer*, *Page 241*.

PEClosePrintJob (Other Declares)

VB Syntax

16-bit

```
Declare Sub PEClosePrintJob Lib "crpe.dll" (ByVal printJob As Integer)
```

32-bit

```
Declare Sub PEClosePrintJob Lib "crpe32.dll" (ByVal printJob As Integer)
```

Delphi Syntax

16-bit

```
function PEClosePrintJob (
    printJob: integer
): Bool;
```

32-bit

```
function PEClosePrintJob (
    printJob: Word
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEClosePrintJob (CWORD) CRPE.DLL
```

PECloseSubreport



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Closes the specified subreport.

```
BOOL CRPE_API PECloseSubreport (
    short printJob//handle to subreport
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the subreport you want to close.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

Once this function has been called, *PEGetSubreportInfo*, *Page 228*, or any other Crystal Report Engine function that applies to the subreport can not be called.

PECloseSubreport (Other Declares)

VB Syntax

16-bit

```
Declare Function PECloseSubreport Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer
```

32-bit

```
Declare Function PECloseSubreport Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PECloseSubreport (  
    printJob: integer  
) : Bool;
```

32-bit

```
function PECloseSubreport (  
    printJob: Word  
) : Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PECloseSubreport(CWORD) CRPE.DLL
```

PECloseWindow

Closes the preview window. Use this function as part of a Custom-Print link to enable the user to review the report in the preview window and then to close the window (in response to a user event).

Use PECloseWindow to replace the Close button when PEShowPrintControls disables the Print Control buttons.

```
void CRPE_API PECloseWindow (  
    short printJob  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to close the preview window.

Return Value

None

Remarks

The preview window will not be closed if it is busy (generating pages, reading records, etc.)

PECloseWindow (Other Declares)

VB Syntax

16-bit

```
Declare Sub PECloseWindow Lib "crpe.dll" (ByVal printJob As Integer)
```

32-bit

```
Declare Sub PECloseWindow Lib "crpe32.dll" (ByVal printJob As Integer)
```

Delphi Syntax

16-bit

```
procedure PECloseWindow (
  printJob: integer
);
```

32-bit

```
procedure PECloseWindow (
  printJob: Word
)stdcall;
```

dBASE for Windows Syntax

```
EXTERN CVOID PECloseWindow (CWORD) CRPE.DLL
```

PEConvertPFIInfoToVInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Converts the value returned in the CurrentValue or DefaultValue member of the *PEParameterFieldInfo*, [Page 427](#), structure to the appropriate type and places the result in a *PEValueInfo*, [Page 466](#), structure.

```
BOOL CRPE_API PEConvertPFIInfoToVInfo(
    void * value,
    short valueType,
    PEValueInfo * valueInfo
);
```

Parameters

Parameter	Description
value	Specifies a pointer to the current or default parameter value (returned from <i>PEGetNthParameterField</i> , Page 197) to convert.
valueType	Specifies the type of parameter field that the value came from. Use one of the following:

Constant	Value	Meaning
PE_PF_NUMBER	0	Number
PE_PF_CURRENCY	1	Currency
PE_PF_BOOLEAN	2	Boolean
PE_PF_DATE	3	Date
PE_PF_STRING	4	String

Parameter	Description
valueInfo	Specifies a pointer to the <i>PEValueInfo</i> , Page 466 , structure that is used to store the value in converted form.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEConvertPFIInfoToVInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEConvertPFIInfoToVInfo Lib "crpe.dll" (ByVal  
value As Any, ByVal valueType As Integer, valueInfo As  
PEValueInfo) As Integer
```

32-bit

```
Declare Function PEConvertPFIInfoToVInfo Lib "crpe32.dll" (ByVal  
value As Any, ByVal valueType As Integer, valueInfo As  
PEValueInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEConvertPFIInfoToVInfo (   
    value: PChar;  
    valueType: Word;  
    var valueInfo: PEValueInfo  
): Bool;
```

32-bit

```
function PEConvertPFIInfoToVInfo (   
    value: PChar;  
    valueType: Word;  
    var valueInfo: PEValueInfo  
): Bool stdcall;
```

PEConvertVInfoToPFIInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Converts a numeric, currency, date, Boolean, or string value contained in *PEValueInfo*, [Page 466](#), into the binary representation expected by *PESetNthParameterField*, [Page 315](#).

```
BOOL CRPE_API PEConvertVInfoToPFIInfo(
    PEValueInfo * valueInfo,
    WORD * valueType,
    void * value
);
```

Parameters

Parameter	Description
valueInfo	Specifies a pointer to <i>PEValueInfo</i> , Page 466 , that contains the value to be converted.
valueType	Specifies a pointer used to store the type of the value being converted which should be assigned to the <i>ValueType</i> member of <i>PEParameterFieldInfo</i> , Page 427 .
value	Specifies a pointer used to store the value in converted form for use in <i>PESetNthParameterField</i> , Page 315 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEConvertVInfoToPFIInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEConvertVInfoToPFIInfo Lib "crpe.dll"
(valueInfo As PEValueInfo, valueType as Integer, ByVal value As
Any) As Integer
```

32-bit

```
Declare Function PEConvertVInfoToPFIInfo Lib "crpe32.dll"
(valueInfo As PEValueInfo, valueType as Integer, ByVal value As
Any) As Integer
```

Delphi Syntax

16-bit

```
function PEConvertVInfoToPFIInfo (
    valueInfo: PEValueInfo;
    valueType: Word;
    value: Pchar
): Bool;
```

32-bit

```
function PEConvertVInfoToPFIInfo (
    var valueInfo: PEValueInfo;
    var valueType: Word;
    value: Pchar
): Bool stdcall;
```

PEDeleteNthGroupSortField

Removes the specified group sort field from the sort order. This function is used as part of a Custom-Print link whenever you want to delete group sort fields that were established for the report at design time. When you give the user the ability to delete group sort field(s) at print time, your link must include code to replace sortFieldN with user-generated values.

This function can be used by itself to delete an existing group sort field when the sort field number is already known.

The function can also be used as one of a series of functions: *PEGetNGroupSortFields*, *Page 180* (called once), *PEGetNthGroupSortField*, *Page 193*, *PEGetHandleString*, *Page 170* (called together as many times as needed to identify the correct sort field), and *PEDeleteNthGroupSortField* (called once, when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then delete an existing group sort field and/or sort order at print time in response to a user selection.

```
BOOL CRPE_API PEDeleteNthGroupSortField (
    short printJob,
    short sortFieldN
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to delete a group sort field.
sortFieldN	Specifies the number of the sort field you want to delete. The first group sort field is field 0. If N = 0, the function will delete the first group sort field. If the report has N group sort fields, the function can be called with sortFieldN between 0 and N-1.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PEDeleteNthGroupSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PEDeleteNthGroupSortField Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal SortFieldN As Integer) As Integer
```

32-bit

```
Declare Function PEDeleteNthGroupSortField Lib "crpe32.dll"  
(ByVal printJob As Integer, ByVal SortFieldN As Integer) As  
Integer
```

Delphi Syntax

16-bit

```
function PEDeleteNthGroupSortField (  
    printJob,  
    sortFieldN: integer  
) : Bool;
```

32-bit

```
function PEDeleteNthGroupSortField (
    printJob: Word;
    sortFieldN: integer
) :Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEDeleteNthGroupSortField (CWORD, CWORD) CRPE.DLL
```

PEDeleteNthSortField

Removes the specified sort field from the sort order. This function is used as part of a Custom-Print link whenever you want to delete sort fields that were established for the report at design time. When you give the user the ability to delete sort field(s) at print time, your link must include code to replace sortFieldN with user-generated values. This function can be used by itself to delete an existing sort field when the sort field number is already known.

The function can also be used as one of a series of functions: *PEGetNSortFields*, *Page 187* (called once), *PEGetNthSortField*, *Page 200*, *PEGetHandleString*, *Page 170* (called together as many times as needed to identify the correct sort field), and *PEDeleteNthSortField* (called once, when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then delete an existing sort field and/or sort order at print time in response to a user selection.

```
BOOL CRPE_API PEDeleteNthSortField (
    short printJob,
    short sortFieldN
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to delete a sort field.
sortFieldN	Specifies the number of the sort fields you want to delete. The first sort field is field 0. If N = 0, the function will delete the first sort field. If the report has N sort fields, you can call the function with sortFieldN between 0 and N-1.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PEDeleteNthSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PEDeleteNthSortField Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal SortFieldN As Integer) As Integer
```

32-bit

```
Declare Function PEDeleteNthSortField Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal SortFieldN As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEDeleteNthSortField ( //  
    printJob,  
    sortFieldN: integer  
): Bool;
```

32-bit

```
function PEDeleteNthSortField ( //  
    printJob: Word;  
    sortFieldN: integer  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEDeleteNthSortField (CWORD, CWORD) CRPE.DLL
```

PEDiscardSavedData

Discards data that was previously saved with the report. If a report has been saved with data, you can use this function to discard the saved data, forcing the Crystal Report Engine to retrieve new data when the report is printed.

```
BOOL CRPE_API PEDiscardSavedData (
    short printJob
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to discard saved data.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PEDiscardSavedData (Other Declares)

VB Syntax

16-bit

```
Declare Function PEDiscardSavedData Lib "crpe.dll" (ByVal
printJob As Integer) As Integer
```

32-bit

```
Declare Function PEDiscardSavedData Lib "crpe32.dll" (ByVal
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEDiscardSavedData (
    printJob: integer
): Bool;
```

32-bit

```
function PEDiscardSavedData (
    printJob: Word
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEDiscardSavedData (CWORD) CRPE.DLL
```



PEEnableEvent



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Enable the capture of Report Engine events.

```
BOOL CRPE_API PEEnableEvent (
    short printJob,
    PEEnableEventInfo Far *enableEventInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to capture events.
enableEventInfo	Specifies a pointer to <i>PEEnableEventInfo</i> , Page 397 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

By default, all events are disabled. Use this function to enable events.

PEEnableEvent (Other Declares)

VB Syntax

16-bit

```
Declare Function PEEnableEvent Lib "crpe.dll" (By Val printJob  
As Integer, enableEventInfo As PEEnableEventInfo) As Integer
```

32-bit

```
Declare Function PEEnableEvent Lib "crpe.dll" (By Val printJob  
As Integer, enableEventInfo As PEEnableEventInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEEnableEvent (  
printJob: integer;  
var enableEventInfo: PEEnableEventInfo  
): integer;
```

32-bit

```
function PEEnableEvent (  
printJob: Word;  
Var enableEventInfo: PEEnableEventInfo  
): Bool stdcall;
```

PEEnableProgressDialog



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Enables/disables the display of the Progress dialog box. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

```
BOOL CRPE_API PEEnableProgressDialog (  
    short printJob,  
    BOOL enable  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to enable/disable the progress dialog box.
enable	Specifies whether or not the progress dialog box is enabled. If enable is set to TRUE (1), the progress dialog box is enabled. If it's set to FALSE (0), the dialog box is disabled.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This dialog box is enabled by default.

PEEnableProgressDialog (Other Declares)

VB Syntax

16-bit

```
Declare Function PEEnableProgressDialog Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal enable As Integer) As Integer
```

32-bit

```
Declare Function PEEnableProgressDialog Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal enable As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEEnableProgressDialog ( //  
  printJob: integer;  
  enable: Bool  
): Bool;
```

32-bit

```
function PEEnableProgressDialog (
    printJob: Word;
    enable: Bool
): Bool stdcall;
```

PEExportPrintWindow

Exports the report displayed in the preview window to disk file or e-mail. This function can be used in a Custom-Print link to enable the user to preview the report in the preview window, and then, if everything looks satisfactory, to export the report to disk file or e-mail (in response to a user event).

Use PEExportPrintWindow to replace the Export button when PEShowPrintControls disables the Print Control buttons.

```
BOOL CRPE_API PEExportPrintWindow (
    short printJob,
    BOOL toMail,
    BOOL waitUntilDone
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to print to a window.
toMail	Indicates whether or not the function is to export to e-mail.
waitUntilDone	Indicates whether or not the function is to return as soon as printing starts. Use one of the following values:

Value	Meaning
TRUE	The function returns when printing is finished. In version 1.1 and higher of CRPE, “waitUntilDone” must be true.
FALSE	The function returns as soon as printing starts.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEExportPrintWindow (Other Declares)

VB Syntax

16-bit

```
Declare Function PEExportPrintWindow Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal ToMail As Integer, ByVal  
WaitUntilDone As Integer) As Integer
```

32-bit

```
Declare Function PEExportPrintWindow Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal ToMail As Integer, ByVal  
WaitUntilDone As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEExportPrintWindow(  
    printJob: integer;  
    toMail: Bool;  
    waitUntilDone: Bool  
): Bool;
```

32-bit

```
function PEExportPrintWindow (  
    printJob: Word;  
    toMail: Bool;  
    waitUntilDone: Bool  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEExportPrintWindow (CWORD, CLOGICAL, CLOGICAL)  
CRPE.DLL
```

PEExportTo

Exports a print job using the name, format, and destination specified in the *PEExportOptions*, *Page 399*. Use this function any time you want to print a report to a file or export it for use in other programs.

```
BOOL CRPE_API PEExportTo (
    short printJob,
    struct PEExportOptions FAR *options
);
```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job that you want to export.
options	Specifies the pointer to <i>PEExportOptions</i> , <i>Page 399</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEExportTo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEExportTo Lib "crpe.dll" (ByVal printJob As
Integer, ExportOptions As PEExportOptions) As Integer
```

32-bit

```
Declare Function PEExportTo Lib "crpe32.dll" (ByVal printJob As
Integer, ExportOptions As PEExportOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEExportTo (
    printJob: integer;
    var options: PEExportOptions
): Bool;
```

32-bit

```
function PEExportTo (
    printJob: Word;
    var options: PEExportOptions
): Bool stdcall;
```

PEGetAreaFormat

Retrieves the area format settings for selected areas in the specified report and supplies them as member values for *PESectionOptions*, *Page 444*. Use this function to update the area formats and pass them back using *PESetAreaFormat*, *Page 269*.

```
BOOL CRPE_API PEGetAreaFormat (
    short printJob,
    short areaCode,
    PESectionOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to get the area format.
areaCode	Specifies the code for the area for which you want to get format information. See <i>Working with section codes</i> , <i>Page 116</i> , for more information.
options	Specifies a pointer to <i>PESectionOptions</i> , <i>Page 444</i> , that will receive format information for the area.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetAreaFormat (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetAreaFormat Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal areaCode As Integer, Options As  
PESectionOptions) As Integer
```

32-bit

```
Declare Function PEGetAreaFormat Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, Options As  
PESectionOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEGetAreaFormat (  
    printJob: integer;  
    areaCode: integer;  
    var options: PESectionOptions  
): Bool;
```

32-bit

```
function PEGetAreaFormat (  
    printJob: Word  
    areaCode: integer;  
    options: PESectionOptions  
): Bool stdcall;
```

PEGetAreaFormatFormula

Retrieves the text of a conditional area format formula as a string handle. Use this function in order to update the conditional area format formula and pass the changes back using PESetAreaFormatFormula.

```
BOOL CRPE_API PEGetAreaFormatFormula (  
    short printJob,  
    short areaCode,
```

```

    short formulaName,
    HANDLE FAR *textHandle,
    short FAR *textLength
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job from which you want to get the area format formula.
areaCode	Specifies the code for the area for which you want to get format information. See <i>Working with section codes, Page 116</i> , for more information.
formulaName	Specifies the name of the formatting formula for which you want to supply a new string. Use one of the following constants:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FFN_AREASECTION _VISIBILITY	58	Area or section invisible.
PE_FFN_SHOW_AREA	59	Area is shown or hidden. When hidden, drill down is OK.
PE_FFN_NEW_PAGE _BEFORE	60	Start a new page before this area.
PE_FFN_NEW_PAGE _AFTER	61	Start a new page after this area.
PE_FFN_KEEP_TOGETHER	62	Keep area together.
PE_FFN_RESET_PAGE_N _AFTER	64	Reset page number after the area.
PE_FFN_PRINT_AT _BOTTOM_OF_PAGE	65	Print at bottom of the page.

<i>Parameter</i>	<i>Description</i>
textHandle	Specifies the pointer to the handle of the string containing the error text.
textLength	Specifies the pointer to the length of the error string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetAreaFormatFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetAreaFormatFormula Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, ByVal  
formulaName As Integer, textHandle As Integer, textLength As  
Integer) As Integer
```

32-bit

```
Declare Function PEGetAreaFormatFormula Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, ByVal  
formulaName As Integer, textHandle As Long, textLength As  
Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetAreaFormatFormula (   
    printJob: Word;  
    areaCode: Word;  
    formulaName: Word;  
    var textHandle: HWnd;  
    var textLength: Word  
): Bool;
```

32-bit

```
function PEGetAreaFormatFormula (   
    printJob: Word;  
    areaCode: Word;  
    formulaName: Word;  
    var textHandle: HWnd;  
    var textLength: Word  
): Bool stdcall;
```



PEGetEnableEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Gets event enable information. Use this function to find out which events are enabled.

```
BOOL CRPE_API PEGetEnableEventInfo (
    short printJob,
    PEEnableEventInfo FAR *enableEventInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to obtain information about enabled events.
enableEventInfo	Specifies a pointer to <i>PEEnableEventInfo</i> , Page 397.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetEnableEventInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetEnableEventInfo Lib "crpe.dll" (ByVal
printJob As Integer, enableEventInfo As PEEnableEventInfo) As
Integer
```

32-bit

```
Declare Function PEGetEnableEventInfo Lib "crpe.dll" (ByVal
printJob As Integer, enableEventInfo As PEEnableEventInfo) As
Integer
```

Delphi Syntax

16-bit

```
function PEGetEnableEventInfo(
    printJob: Integer,
    var enableEventInfo: PEEnableEventInfo
): Integer;
```

32-bit

```
function PEGetEnableEventInfo(
    printJob: Word;
    Var enableEventInfo: PEEnableEventInfo
): Bool stdcall;
```

PEGetErrorCode

Returns a number that indicates the status of the most recent Crystal Report Engine function called. When a call to another function fails, this call gets the error code that was generated so you can take some action based on that error code.

```
short CRPE_API PEGetErrorCode (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve an error code.

Return Value

Returns 0 if the function completed without error; returns the code number of the most recent error for the given job if the function was unsuccessful. (Search for *Crystal Report Engine Error Codes* Developer's online Help.)

Remarks

If the most recent function was called while no print job was open (i.e., PEOpenEngine), use 0 for 'printJob'. PEGetErrorCode must be called immediately after the call to the function which indicated an error.

PEGetErrorCode (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetErrorCode Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer
```

32-bit

```
Declare Function PEGetErrorCode Lib "crpe32.dll" (ByVal printJob  
As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetErrorCode (  
    printJob: integer  
) : integer;
```

32-bit

```
function PEGetErrorCode (  
    printJob: Word  
) : Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetErrorCode (CWORD) CRPE.DLL
```

PEGetErrorText

Returns a string handle describing the status of the most recent Crystal Report Engine function called. This function is typically used together with *PEGetHandleString*, *Page 170*. These functions can be used in a Custom-Print link to display the error string to the user as part of an error message.

```
BOOL CRPE_API PEGetErrorText (  
    short printJob,  
    HANDLE FAR *textHandle,  
    short Far *textLength  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve an error string.
textHandle	Specifies the pointer to the handle of the string containing the error text.
textLength	Specifies the pointer to the length of the error string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

If the most recent function was called while no print job was open, use 0 for 'printJob'.

PEGetErrorText (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetErrorText Lib "crpe.dll" (ByVal printJob  
As Integer, TextHandle As Integer, TextLength As Integer) As  
Integer
```

32-bit

```
Declare Function PEGetErrorText Lib "crpe32.dll" (ByVal printJob  
As Integer, TextHandle As Long, TextLength As Integer) As  
Integer
```

Delphi Syntax

16-bit

```
function PEGetErrorText (  
    printJob: integer;  
    var textHandle: HWnd;  
    var textLength: integer  
): Boolean;
```

32-bit

```
function PEGetErrorText (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
): Bool stdcall;
```

PEGetExportOptions

Gets export options from the user before exporting the report. PEGetExportOptions can be used to present a series of dialog boxes that retrieve export options from your users. These options are used by the Crystal Report Engine to fill in the PEEportOptions structure. The *PEExportTo*, *Page 146* function can then be used to set the print job destination using the information in PEEportOptions.

```
BOOL CRPE_API PEGetExportOptions (
    short printJob,
    PEEportOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the print job from which you want to get export options.
options	Specifies the pointer to <i>PEExportOptions</i> , <i>Page 399</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetExportOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetExportOptions Lib "crpe.dll" (ByVal
printJob As Integer, ExportOptions As PEEportOptions) As
Integer
```

32-bit

```
Declare Function PEGetExportOptions Lib "crpe32.dll" (ByVal  
printJob As Integer, ExportOptions As PEExportOptions) As  
Integer
```

Delphi Syntax

16-bit

```
function PEGetExportOptions (  
    printJob: integer;  
    var options: PEExportOptions  
): Bool;
```

32-bit

```
function PEGetExportOptions (  
    printJob: Word;  
    var options: PEExportOptions  
): Bool stdcall;
```

PEGetFormula

Returns the text of the named formula as a string handle. This function is typically used as one of a series of functions (PEGetFormula, *PEGetHandleString*, *Page 170*, *PESetFormula*, *Page 287*). The series can be used in a Custom-Print link to identify and then change an existing formula at print time in response to a user selection.

```
BOOL CRPE_API PEGetFormula (  
    short printJob,  
    char *formulaName,  
    HANDLE FAR *textHandle,  
    short FAR *textLength  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve the formula string.
formulaName	Specifies a pointer to the null-terminated string that contains the name of the formula for which you want to retrieve the formula string.

<i>Parameter</i>	<i>Description</i>
textHandle	Specifies the pointer to the handle of the string containing the formula text.
textLength	Specifies the pointer to the length of the formula string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the named formula does not exist in the report.

PEGetFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetFormula Lib "crpe.dll" (ByVal printJob As Integer, ByVal FormulaName As String, TextHandle As Integer, TextLength As Integer) As Integer
```

32-bit

```
Declare Function PEGetFormula Lib "crpe32.dll" (ByVal printJob As Integer, ByVal FormulaName As String, TextHandle As Long, TextLength As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetFormula (
  printJob: integer;
  formula name: PChar;
  var textHandle: HWnd;
  var textLength: integer
): Boolean;
```

32-bit

```
function PEGGetFormula (
    printJob: Word;
    formulaName: PChar;
    var textHandle: HWnd;
    var textLength: Word
): Bool stdcall;
```

PEGetGraphData

Identifies the data on which the specified graph in the specified section is based. Use this function to find out what data in your report is being used by a specified graph when the graph is created. This information includes which groups are used to create the rows and columns of the graph and which summary field in the report is used to set the values of the risers in the graph.

```
BOOL CRPE_API PEGetGraphData (
    short printJob,
    short sectionCode,
    short graphN,
    PEGraphDataInfo FAR *graphDataInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to get information about a graph.
sectionCode	Specifies the code for the report section in which the graph appears. See the table of section constants supplied with <i>PESetSectionFormat</i> , <i>Page 331</i> .
graphN	Specifies which graph within the section you want data for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphDataInfo	Specifies the pointer to <i>PEGraphDataInfo</i> , <i>Page 410</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetGraphData (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetGraphData Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphDataInfo As PEGraphDataInfo) As Integer
```

32-bit

```
Declare Function PEGetGraphData Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphDataInfo As PEGraphDataInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEGetGraphData (  
    printJob: integer;  
    sectionCode: integer;  
    graphN: integer;  
    var graphDataInfo:PEGraphDataInfo  
): Bool;
```

32-bit

```
function PEGetGraphData (  
    printJob: Word;  
    sectionCode: integer;  
    graphN: integer;  
    var graphDataInfo:PEGraphDataInfo  
): Bool stdcall;
```

PEGetGraphOptions

Identifies the display options that are set for the specified graph in the specified section. Use this function to obtain information about any of several graph options. These options include the minimum and maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

```

BOOL CRPE_API PEGetGraphOptions (
    short printJob,
    short sectionCode,
    short graphN,
    PEGraphOptions FAR *graphOptions
) ;

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job from which you want to get information about a graph.
sectionCode	Specifies the code for the report section in which the graph appears. See the table of section constants supplied with <i>PESetSectionFormat</i> , <i>Page 331</i> .
graphN	Specifies which graph within the section you want information about. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphOptions	Specifies the pointer to <i>PEGraphOptions</i> , <i>Page 413</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetGraphOptions (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPEGetGraphOptions in Developer's online Help.

Delphi Syntax

16-bit

```

function PEGetGraphOptions (
    printJob: integer;
    sectionCode: integer;
    graphN: integer;
    var graphOptions:PEGraphOptions
    ): Bool;

```

32-bit

```
function PEGetGraphOptions (
    printJob: Word;
    sectionCode: integer;
    graphN: integer;
    var graphOptions:PEGraphOptions
): Bool stdcall;
```

PEGetGraphText

Gets the identifying text that appears on the specified graph in the specified section. This function allows you to find out what text appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs).

```
BOOL CRPE_API PEGetGraphText (
    short printJob,
    short sectionCode,
    short graphN,
    PEGraphTextInfo FAR *graphTextInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to get the text displayed with a graph.
sectionCode	Specifies the code for the report section in which the graph appears. See the table of section constants supplied with <i>PESetSectionFormat</i> , <i>Page 331</i> .
graphN	Specifies which graph within the section you want the text for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphTextInfo	Specifies the pointer to <i>PEGraphTextInfo</i> , <i>Page 415</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetGraphText (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetGraphText Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphTextInfo As PEGraphTextInfo) As Integer
```

32-bit

```
Declare Function PEGetGraphText Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphTextInfo As PEGraphTextInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEGetGraphText (  
    printJob: integer;  
    sectionCode: integer;  
    graphN: integer;  
    var graphTextInfo:PEGraphTextInfo  
): Bool;
```

32-bit

```
function PEGetGraphText (  
    printJob: Word;  
    sectionCode: integer;  
    graphN: integer;  
    var graphTextInfo:PEGraphTextInfo  
): Bool stdcall;
```

PEGetGraphType

Identifies one of the available graph/chart types used for the specified graph in the specified section. Use this function to find out what type of graph is being displayed in the report. There are many types of graphs and charts possible with Seagate Crystal Reports. This function will return the type of graph or chart being displayed in a report.

```

BOOL CRPE_API PEGetGraphType (
    short printJob,
    short sectionCode,
    short graphN,
    short FAR *graphType
) ;

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job from which you want to find out the type of graph being displayed.
sectionCode	Specifies the code for the report section in which the graph appears. See the table of section constants supplied with <i>PESetSectionFormat</i> , <i>Page 331</i> .
graphN	Specifies which graph within the section you want to know the type of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphType	Returns a pointer to the type of graph being displayed. See <i>PESetGraphType</i> , <i>Page 294</i> , for the list of possible graph types.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetGraphType (Other Declares)

VB Syntax

16-bit

```

Declare Function PEGetGraphType Lib "crpe.dll" (ByVal printJob
As Integer, ByVal sectionCode As Integer, ByVal GraphN As
Integer, GraphType As Integer) As Integer

```

32-bit

```

Declare Function PEGetGraphType Lib "crpe32.dll" (ByVal printJob
As Integer, ByVal sectionCode As Integer, ByVal GraphN As
Integer, GraphType As Integer) As Integer

```

Delphi Syntax

16-bit

```
function PEGetGraphType (
    printJob: integer;
    sectionCode: integer;
    graphN: integer;
    var graphType: integer
) :Bool;
```

32-bit

```
function PEGetGraphType (
    printJob: Word;
    sectionCode: integer;
    graphN: integer;
    var graphType: Word
) : Bool stdcall;
```

PEGetGroupCondition



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Determines the group condition information for a selected group section in the specified report. Use this function to find out the group condition for a group section. Use *PESetGroupCondition*, Page 296, to change the group condition once it is known.

```
BOOL CRPE_API PEGetGroupCondition (
    short printJob,
    short sectionCode,
    HANDLE FAR *conditionFieldHandle,
    short FAR *conditionFieldLength,
    short FAR *condition,
    short FAR *sortDirection
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to determine the group conditions for a selected group.

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the report section for which you want to get the group condition. See the table of section constants supplied with <i>PESetSectionFormat</i> , <i>Page 331</i> .
conditionFieldHandle	Specifies a pointer to the handle of the condition field for the selected group section.
conditionFieldLength	Specifies a pointer to the length of the condition field for the selected group section.
condition	Specifies a pointer to the condition setting for the selected group section (date and Boolean fields only). See <i>PESetGroupCondition</i> , <i>Page 296</i> , for information on the condition constants.
sortDirection	Specifies a pointer to the sort direction setting for the selected group section. See <i>PESetGroupCondition</i> , <i>Page 296</i> , for information on the sort direction constants.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

The *condition* parameter returns a value that encodes both the condition and the type of the condition field. You need to apply a condition mask (PE_GC_CONDITIONMASK) or a type mask (PE_GC_TYPEMASK) against this value using a bitwise AND to determine the condition or type respectively. For example:

```
short result;

result = *condition & PE_GC_TYPEMASK;

if (result == PE_GC_TYPEDATE)
{
    //what you want it to do
}
```

Type can be any of the following:

- **PE_GC_TYPEOTHER**
Any data type other than date or Boolean.
- **PE_GC_TYPEDATE**
Date data type.
- **PE_GC_TYPEBOOLEAN**
Boolean data type.

PEGetGroupCondition (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetGroupCondition Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer,  
ConditionFieldHandle As Integer, ConditionFieldLength As  
Integer, Condition As Integer, SortDirection As Integer) As  
Integer
```

32-bit

```
Declare Function PEGetGroupCondition Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer,  
ConditionFieldHandle As Long, ConditionFieldLength As Integer,  
Condition As Integer, SortDirection As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetGroupCondition (   
    printJob: integer;  
    sectionCode: integer;  
    var conditionFieldHandle: HWnd;  
    var conditionFieldLength: integer;  
    var condition: integer;  
    var sortDirection: integer  
 ): Bool;
```

32-bit

```
function PEGetGroupCondition (   
    printJob: Word;  
    sectionCode: integer;  
    var conditionFieldHandle: Hwnd;  
    var conditionFieldLength: Word;  
    var condition: Word;  
    var sortDirection: Word  
 ): Bool stdcall;
```

PEGetGroupOptions

Retrieves the current setting for specified groups in your report.

```
BOOL CRPE_API PEGetGroupOptions (
    short printJob,
    short groupN,
    PEGroupOptions FAR *groupOptions
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you wish to obtain information about the group options for a specific group.
groupN	Specifies the group number. The first level group is 0.
groupOptions	Specifies the pointer to <i>PEGroupOptions</i> , <i>Page 418</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function gets all the information *PEGetGroupCondition*, *Page 164*, can get plus additional group options such as repeat group header, keep group together, and information about top/bottom n group sorting information.

PEGetGroupOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetGroupOptions Lib "crpe.dll" (ByVal
printJob As Integer, ByVal groupN As Integer, groupOptions As
PEGroupOptions) As Integer
```

32-bit

```
Declare Function PEGetGroupOptions Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal groupN As Integer, groupOptions As  
PEGroupOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEGetGroupOptions  
    printJob: Word;  
    group N: smallint;  
    var groupOptions: PEGroupOptions  
): Bool;
```

32-bit

```
function PEGetGroupOptions  
    printJob: Word;  
    groupN: smallint;  
    var groupOptions: PEGroupOptions  
): integer; stdcall;
```

PEGetGroupSelectionFormula

Returns the string handle for the group selection formula used in the specified report. This function is typically used as one of a series of functions (PEGetGroupSelectionFormula, *PEGetHandleString*, *Page 170*, *PESetGroupSelectionFormula*, *Page 303*). This series can be used in a custom-print link to identify and then change an existing group selection formula at print time in response to a user selection.

```
BOOL CRPE_API PEGetGroupSelectionFormula (  
    short printJob,  
    HANDLE FAR *textHandle,  
    short FAR *textLength  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to retrieve the group selection formula string.

<i>Parameter</i>	<i>Description</i>
textHandle	Specifies the pointer to the handle of the string containing the formula text.
textLength	Specifies the pointer to the length of the formula string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetGroupSelectionFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetGroupSelectionFormula Lib "crpe.dll"
    (ByVal printJob As Integer, TextHandle As Integer, TextLength As
    Integer) As Integer
```

32-bit

```
Declare Function PEGetGroupSelectionFormula Lib "crpe32.dll"
    (ByVal printJob As Integer, TextHandle As Long, TextLength As
    Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetGroupSelectionFormula (
    printJob: integer;
    var textHandle: HWnd;
    var textLength: integer
): Boolean;
```

32-bit

```
function PEGetGroupSelectionFormula (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
): Boolean stdcall;
```

PEGetHandleString

Returns the text that the string handle is pointing to. The buffer will obtain the actual text. This function is used in conjunction with functions that return variable length strings. After your program allocates a buffer of sufficient size, this function moves the string from the string handle to the buffer.

```
BOOL CRPE_API PEGetHandleString (
    HANDLE textHandle,
    char FAR *buffer,
    short bufferLength
);
```

Parameters

Parameter	Description
textHandle	Specifies the handle to the string containing the text of interest. Obtained from a variable length string function.
buffer	Specifies a pointer to the buffer into which you want the string copied.
bufferLength	Specifies the length of the buffer in bytes, including the terminating null byte. This value should be identical to the length of the string obtained by the variable length string function.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- You can only use this call once with a given string handle because the string handle is discarded once the function is called. If you expect to use the string later, you will need to save it.
- PEGetHandleString will copy, at most, the number of bytes indicated by bufferLength, ensuring that the string in the buffer is null-terminated.

NOTE: When you call the function that produces the string, it returns a length that includes a provision for the null byte at the end of the string. A buffer set to that length will hold the entire string including the terminating null byte.

PEGetHandleString (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crvbHandleToBStr in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetHandleString (
    textHandle: HWnd;
    buffer: PChar;
    bufferLength: integer
): Bool;
```

32-bit

```
function PEGetHandleString (
    textHandle: HWnd;
    buffer: PChar;
    bufferLength: integer
): Bool stdcall;
```

PEGetJobStatus

Evaluates the status of a print job. You can use this function in a number of programming situations, for example:

- to trigger error messages, i.e., when a print job fails (due to insufficient memory, insufficient disk space, etc.),
- to trigger screen displays (hourglass, series of graphics, etc.) that confirm to the user that work is in progress, or
- to find out whether a job was cancelled by the user after *PEStartPrintJob*, *Page 348*, returns.

```
short CRPE_API PEGetJobStatus (
    short printJob,
    struct PEJobInfo FAR *jobInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query about its printing status.
jobInfo	Specifies the pointer to <i>PEJobInfo</i> , <i>Page 423</i> , that will capture the information this function retrieves.

Return Value

- “0” if *PEOpenEngine*, *Page 247*, or *PEOpenPrintJob*, *Page 248*, has not been called successfully.
- “1” if the job has not started yet (PE_JOBNOTSTARTED).
- “2” if the job is in progress (PE_JOBINPROGRESS).
- “3” if the job has completed successfully (PE_JOBCOMPLETED).
- “4” if the job has failed (PE_JOBFAILED).
- “5” if the job has been canceled by the user (PE_JOBCANCELLED).
- “6” if the report has exceeded the record or time governors (PE_JOBHALTED).

PEGetJobStatus (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetJobStatus Lib "crpe.dll" (ByVal printJob  
As Integer, JobInfo As PEJobInfo) As Integer
```

32-bit

```
Declare Function PEGetJobStatus Lib "crpe32.dll" (ByVal printJob  
As Integer, JobInfo As PEJobInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEGetJobStatus (
    printJob: integer;
    var jobInfo: PEJobInfo
): integer;
```

32-bit

```
function PEGetJobStatus (
    printJob: Word;
    var jobInfo: PEJobInfo
): smallint stdcall;
```

PEGetMargins



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the page margin settings for the specified report. Use this function to find out what the currently set margins for the report are. Use this function with PESetMargins. Use PEGetMargins to obtain report margins and use *PESetMargins*, Page 305, to set the report margins.

```
BOOL CRPE_API PEGetMargins (
    short printJob,
    short FAR *left,
    short FAR *right,
    short FAR *top,
    short FAR *bottom
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to retrieve margin information.
left	Specifies a pointer to the value of the left margin.
right	Specifies a pointer to the value of the right margin.
top	Specifies a pointer to the value of the top margin.
bottom	Specifies a pointer to the value of the bottom margin.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetMargins (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetMargins Lib "crpe.dll" (ByVal printJob As Integer, LeftMargin As Integer, RightMargin As Integer, TopMargin As Integer, BottomMargin As Integer) As Integer
```

32-bit

```
Declare Function PEGetMargins Lib "crpe32.dll" (ByVal printJob As Integer, LeftMargin As Integer, RightMargin As Integer, TopMargin As Integer, BottomMargin As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetMargins (
    printJob: integer;
    var left: Word;
    var right: Word;
    var top: Word;
    var bottom: Word
): Bool;
```

32-bit

```
function PEGetMargins (
    printJob: Word;
    var left: Word;
    var right: Word;
    var top: Word;
    var bottom: Word
): Bool stdcall;
```

PEGetMinimumSectionHeight



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the minimum section height information for selected sections in the specified report. The minimum section height is the height that can contain all its objects.

```
BOOL CRPE_API PEGetMinimumSectionHeight (
    short printJob,
    short sectionCode,
    short FAR *minimumHeight
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to retrieve information on minimum section height.
sectionCode	Specifies the code for the report section(s) for which you want to retrieve information on minimum section height. See <i>Working with section codes, Page 116</i> .
minimumHeight	Specifies a pointer to the minimum height (in twips) you want to set for the selected section(s).

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetMinimumSectionHeight (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetMinimumSectionHeight Lib "crpe.dll" (ByVal
printJob As Integer, ByVal sectionCode As Integer, MinimumHeight
As Integer) As Integer
```

32-bit

```
Declare Function PEGetMinimumSectionHeight Lib "crpe32.dll"
(ByVal printJob As Integer, ByVal sectionCode As Integer,
MinimumHeight As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetMinimumSectionHeight (
    printJob: integer;
    sectionCode: integer;
    var minimumHeight: integer
): Bool;
```

32-bit

```
function PEGetMinimumSectionHeight (
    printJob: Word;
    sectionCode: integer;
    var minimumHeight: Word
): Bool stdcall;
```

PEGetNDetailCopies



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Returns the number of copies of each Details section in the report that are to be printed. Use this function to find out how many times each Details section of the report will be printed. To change the number of times each Details section is printed, use *PESetNDetailCopies*, *Page 309*.

```
BOOL CRPE_API PEGetNDetailCopies (
    short printJob,
    short FAR *nDetailCopies
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to know how many times each Details section is to be printed.
nDetailCopies	Holds the number of copies of the Details section to be printed.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetNDetailCopies (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNDetailCopies Lib "crpe.dll" (ByVal  
printJob As Integer, nDetailCopies As Integer) As Integer
```

32-bit

```
Declare Function PEGetNDetailCopies Lib "crpe32.dll" (ByVal  
printJob As Integer, nDetailCopies As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNDetailCopies ( //  
    printJob: integer;  
    var nDetailCopies: integer  
): Bool;
```

32-bit

```
function PEGetNDetailCopies( //  
    printJob: Word;  
    var nDetailCopies: integer  
): Bool stdcall;
```

PEGetNFormulas

Determines the number of formulas in the specified report. Use this function to fetch the number of formulas in the report. To fetch the formula by number, use *PEGetNthFormula*, *Page 191*.

```
short CRPE_API PEGetNFormulas ( //  
    short printJob  
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to find the number of formulas it contains.

Return Value

The number of formulas in the report. Returns -1 if an error occurs.

PEGetNFormulas (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNFormulas Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer
```

32-bit

```
Declare Function PEGetNFormulas Lib "crpe32.dll" (ByVal printJob  
As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNFormulas (  
  printJob: integer  
) : Bool;
```

32-bit

```
function PEGetNFormulas (  
  printJob: Word  
) : Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNFormulas (CWORD) CRPE.DLL
```

PEGetNGroups



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Determines the number of groups in the specified report.

```
short CRPE_API PEGetNGroups (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to determine the number of groups it contains.

Return Value

Returns the number of groups in the report. Returns -1 if an error occurs.

PEGetNGroups (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNGroups Lib "crpe.dll" (ByVal printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNGroups Lib "crpe32.dll" (ByVal printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNGroups (
    printJob: integer
): integer;
```

32-bit

```
function PEGetNGroups (
    printJob: Word
): smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNGroups (CWORD) CRPE.DLL
```

PEGetNGroupSortFields

Returns the number of group sort fields in the specified report. This function is typically used as one of a series of functions: *PEGetNGroupSortFields* (called once), *PEGetNthGroupSortField*, *Page 193*, *PEGetHandleString*, *Page 170* (called as many times as needed to identify the correct group sort field), and *PESetNthGroupSortField*, *Page 311*, (called once, when the correct group sort field is identified). The series can be used in a custom-print link to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

NOTE: A group sort field is a summary value or a summarized field that determines the sort order of groups.

```
short CRPE_API PEGetNGroupSortFields (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to find out how many group sort fields it contains.

Return Value

If group sort fields have been defined, returns the number of such fields; if there are no group sort fields defined, returns 0; if an error occurs, returns -1.

PEGetNGroupSortFields (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNGroupSortFields Lib "crpe.dll" (ByVal  
printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNGroupSortFields Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNGroupSortFields (   
    printJob: integer  
): integer;
```

32-bit

```
function PEGetNGroupSortFields (   
    printJob: Word  
): Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNGroupSortFields (CWORD) CRPE.DLL
```

PEGetNPages



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the number of pages in the report.

```
short CRPE_API PEGetNPages (   
    short printJob  
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to determine the number of pages.

Return Value

The number of pages in the report.

PEGetNPages (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNPages Lib "crpe.dll" (ByVal printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNPages Lib "crpe32.dll" (ByVal printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNPages (
  printJob: integer
): integer;
```

32-bit

```
function PEGetNPages (
  printJob: Word
): Smallint stdcall;
```

PEGetNParameterFields



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Determines the number of parameter fields used in the report, including the parameter fields in all the subreports.

```
short CRPE_API PEGetNParameterFields (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve a parameter field count.

Return Value

The number of parameter fields in the report. Returns -1 if an error occurs.

PEGetNParameterFields (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNParameterFields Lib "crpe.dll" (ByVal
printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNParameterFields Lib "crpe32.dll" (ByVal
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNParameterFields (
    printJob: integer
): integer;
```

32-bit

```
function PEGetNParameterFields (
    printJob: Word
): Smallint stdcall;
```

PEGetNParams



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the number of parameters needed by a stored procedure. Use this function whenever you need to know how many parameters are required by a stored procedure in a SQL database table. This function is usually used in conjunction with *PEGetNthParam*, *Page 195*, or *PESetNthParam*, *Page 313*.

```
short CRPE_API PEGetNParams (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job containing the stored procedure you need to query to find out the number of parameters required by the procedure.

Return Value

The number of parameters used by a stored procedure. Returns -1 if an error occurs.

Related Menu Commands

Database | Stored Procedure Parameters

PEGetNParams (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNParams Lib "crpe.dll" (ByVal printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNParams Lib "crpe32.dll" (ByVal printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNParams (
  printJob: integer
): integer;
```

32-bit

```
function PEGetNParams (
  printJob: Word
): smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNParams (CWORD) CRPE.DLL
```

PEGetNSections

Retrieves the number of sections in the specified report. By default, each report has five areas, each containing one section (Report Header, Page Header, Details, Report Footer, and Page Footer). Thus, if this function were applied to a default report, the return value would be five (5). As you add groups to your report or you add sections to one or more areas, the number of sections in the report increases.

```
short CRPE_API PEGetNSections (
  short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to get a section count.

Return Value

The number of sections in the report. Returns -1 if an error occurs.

Remarks

For information on section codes, see *Working with section codes, Page 116*.

PEGetNSections (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNSections Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer
```

32-bit

```
Declare Function PEGetNSections Lib "crpe32.dll" (ByVal printJob  
As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNSections (  
    printJob: integer  
) : integer;
```

32-bit

```
function PEGetNSections (  
    printJob: Word  
) : Smallint stdcall;
```

PEGetNSortFields

Returns the number of sort fields in the specified report. This function is typically used as one of a series of functions: PEGetNSortFields (called once), *PEGetNthSortField*, *Page 200*, *PEGetHandleString*, *Page 170* (called together as many times as needed to identify the correct sort field), *PESetNthSortField*, *Page 316*, (called once when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then change an existing sort field and/or sort order at print time in response to a user selection.

```
short CRPE_API PEGetNSortFields (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to find out how many sort fields it contains.

Return Value

If there are sort fields defined, returns the number of sort fields; if there are no sort fields defined, returns 0; if an error occurs, returns -1.

Related Menu Commands

Report | Record Sort Order

PEGetNSortFields (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNSortFields Lib "crpe.dll" (ByVal printJob
As Integer) As Integer
```

32-bit

```
Declare Function PEGetNSortFields Lib "crpe32.dll" (ByVal
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNSortFields (
    printJob: integer
): integer;
```

32-bit

```
function PEGetNSortFields (
    printJob: Word
): smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNSortFields (CWORD) CRPE.DLL
```

PEGetNSubreportsInSection



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Determines the number of subreports in the specified section.

```
short CRPE_API PEGetNSubreportsInSection (
    short printJob,
    short sectionCode
);
```

Parameters

Parameter	Description
print job	Specifies the handle of the primary report from which you want to fetch information about the number of subreports in a section.
sectionCode	Specifies the section code of the section for which you want a subreport count. See <i>Working with section codes, Page 116</i> .

Return Value

The number of subreports in the specified section. Returns -1 if an error occurs.

Remarks

sectionCode can be retrieved using *PEGetSectionCode*, *Page 215*.

PEGetNSubreportsInSection (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNSubreportsInSection Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer) As Integer
```

32-bit

```
Declare Function PEGetNSubreportsInSection Lib "crpe32.dll"  
(ByVal printJob As Integer, ByVal sectionCode As Integer) As  
Integer
```

Delphi Syntax

16-bit

```
function PEGetNSubreportsInSection (   
    printJob: integer;  
    sectionCode: integer  
): Integer;
```

32-bit

```
function PEGetNSubreportsInSection (   
    printJob: Word;  
    sectionCode: Smallint  
): Smallint stdcall;
```

PEGetNTables

Retrieves the number of tables in the open report. It counts both PC and SQL databases.

This function is one of a series of functions that enable you to retrieve and update database information in an opened report so that the report can be printed using different server, database, user, and/or table location settings.

```
short CRPE_API PEGetNTables (
    short printJob
);
```

Parameters

Parameter	Description
print job	Specifies the handle of the print job from which you want to retrieve a table count.

Return Value

Returns the number of tables used in the report (1 = 1 table, 2 = 2 tables, etc.). Returns -1 if an error occurs.

Remarks

This function can be used with all compatible PC databases (i.e., Paradox, Xbase) as well as SQL databases (i.e., SQL Server, Oracle, Netware).

PEGetNTables (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNTables Lib "crpe.dll" (ByVal printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetNTables Lib "crpe32.dll" (ByVal printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNTables (
    printJob: Integer
): Integer;
```

32-bit

```
function PEGetNTables (
    printJob: Word
): smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetNTables (CWORD) CRPE.DLL
```

PEGetNthFormula

Retrieves information about a specific formula in the report. Use this function to obtain the formula name and formula text of a specific formula in the report. This function can be used to get formula text, allow the user to edit the formula, then change the formula text with *PESetFormula*, *Page 287*. The first formula is formula 0.

```
BOOL CRPE_API PEGetNthFormula (
    short printJob,
    short formulaN,
    HANDLE FAR *nameHandle,
    short FAR *nameLength,
    HANDLE FAR *textHandle,
    short FAR *textLength
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to gather formula information.
formulaN	Specifies the number of the formula about which you want to gather information.
nameHandle	Specifies a pointer to the handle of the string containing the formula name.
nameLength	Specifies a pointer to the length of the formula name string (in bytes) including the terminating byte.
textHandle	Specifies a pointer to the handle of the string containing the formula text.
textLength	Specifies a pointer to the length of the formula string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Insert | Formula Field

Edit | Formula

PEGetNthFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthFormula Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal formulaN As Integer, NameHandle As Integer,  
NameLength As Integer, TextHandle As Integer, TextLength As  
Integer) As Integer
```

32-bit

```
Declare Function PEGetNthFormula Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal formulaN As Integer, NameHandle As  
Long, NameLength As Integer, TextHandle As Long, TextLength As  
Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthFormula (  
    printJob: integer;  
    formulaN: integer;  
    var nameHandle: HWnd;  
    var nameLength: integer;  
    var textHandle: HWnd;  
    var textLength: integer  
): Boolean;
```

32-bit

```
function PEGetNthFormula (
    printJob: Word;
    formulaN: integer;
    var nameHandle: Hwnd;
    var nameLength: Word;
    var textHandle: Hwnd;
    var textLength: Word
): Bool stdcall;
```

PEGetNthGroupSortField

Returns information about one of the group sort fields in the specified report: that is, it returns the name of the field and the direction (ascending or descending) of the sort. The name of the group sort field is returned as a string handle. See *Crystal Report Engine API variable length strings, Page 63*, for further information.

This function is typically used as one of a series of functions: *PEGetNGroupSortFields, Page 180* (called once), *PEGetNthGroupSortField, PEGetHandleString, Page 170* (called as many times as needed to identify the correct group sort field), and *PESetNthGroupSortField, Page 311* (called once when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

NOTE: A group sort field is a summary value or a summarized field that determines the sort order of groups.

```
BOOL CRPE_API PEGetNthGroupSortField (
    short printJob,
    short sortFieldN,
    HANDLE FAR *nameHandle,
    short FAR *nameLength,
    short FAR *direction
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to gather group sort field information.
sortFieldN	Specifies the number of the group sort field you want to retrieve. The first group sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1.

Parameter	Description
nameHandle	Specifies the pointer to the handle of the string containing the sort field name.
nameLength	Specifies the pointer to the length of the field name string (in bytes) including the terminating byte.
direction	Specifies a pointer to the sort direction. See <i>PESetNthGroupSortField</i> , Page 311, for a list of sort direction constants.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

To find out top/bottom n group sort information, use *PEGetGroupOptions*, Page 167.

Related Menu Commands

Report | Group Sort Order

PEGetNthGroupSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthGroupSortField Lib "crpe.dll" (ByVal
printJob As Integer, ByVal SortFieldN As Integer, NameHandle As
Integer, NameLength As Integer, Direction As Integer) As Integer
```

32-bit

```
Declare Function PEGetNthGroupSortField Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal SortFieldN As Integer, NameHandle As
Long, NameLength As Integer, Direction As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthGroupSortField (
    printJob,
    sortFieldN: integer;
    var nameHandle: HWnd;
    var nameLength: integer;
    var direction: integer
): Boolean;
```

32-bit

```
function PEGetNthGroupSortField (
    printJob: Word;
    sortFieldN: integer;
    var nameHandle: HWnd;
    var nameLength: Word;
    var direction: Word
): Boolean stdcall;
```

PEGetNthParam



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Gets the Nth parameter of a stored procedure. Use this function whenever you need to find out a particular parameter required by a stored procedure in a SQL database table.

```
BOOL CRPE_API PEGetNthParam (
    short printJob,
    short paramN,
    HANDLE FAR *textHandle,
    short FAR *textLength
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job that contains the stored procedure for which you are interested in obtaining a parameter.

<i>Parameter</i>	<i>Description</i>
paramN	0 indexed parameter number, indicates which parameter in the stored procedure you wish to know.
textHandle	Specifies the pointer to the handle of the string containing the sort procedure parameter.
textLength	Specifies the pointer to the length of the parameter string (in bytes) including the terminating null byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function is usually used with *PEGetHandleString*, *Page 170*, to obtain a text string containing the parameter in a stored procedure.
- Use *PEGetNParams*, *Page 184*, to find out the number of parameters required by a stored procedure.
- Use *PESetNthParam*, *Page 313*, to set the value of a parameter in a stored procedure.
- If the current value of the parameter in the stored procedure is NULL, then after this function is called, textHandle will contain NULL and textLength will contain 0.

Related Menu Commands

Database | Stored Procedure Parameters

PEGetNthParam (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthParam Lib "crpe.dll" (ByVal printJob As Integer, ByVal ParamN As Integer, TextHandle As Integer, TextLength As Integer) As Integer
```

32-bit

```
Declare Function PEGetNthParam Lib "crpe32.dll" (ByVal printJob As Integer, ByVal paramN As Integer, TextHandle As Long, TextLength As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthParam (
    printJob: integer;
    paramN: integer;
    var textHandle: HWnd;
    var textLength: integer
): Boolean;
```

32-bit

```
function PEGetNthParam(
    printJob: Word;
    paramN: integer;
    var textHandle: HWnd;
    var textLength: Word
): Boolean stdcall;
```

PEGetNthParameterField



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Returns information about one of the parameter fields in the specified report: that is, it returns the name of the field, the data type, and information about the value set for the field. The name of the parameter field is returned as a string handle. This function is typically used as one of a series of functions: *PEGetNParameterFields*, *Page 183* (called once), *PEGetNthParameterField* (called as many times as needed to identify the correct parameter field), and *PESetNthParameterField*, *Page 315* (called once when the correct parameter field is identified). The series can be used in a Custom-Print link to identify and then change an existing parameter field value at print time in response to a user selection.

```
BOOL CRPE_API PEGetNthParameterField (
    short printJob,
    short parameterN,
    PEPParameterFieldInfo FAR *parameterInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job that contains the parameter field about which you want to retrieve information.
parameterN	Specifies the number of the parameter field about which you want to retrieve information.
parameterInfo	Specifies a pointer to <i>PEParameterFieldInfo</i> , Page 427, that is used to store the information you retrieve.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetNthParameterField (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPEGetNthParameterField in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetNthParameterField (
    printJob: integer;
    varN: integer;
    var varInfo: PEParameterFieldInfo
): Bool;
```

32-bit

```
function PEGetNthParameterField (
    printJob: Word;
    varN: Smallint;
    var varInfo: PEParameterFieldInfo
): Bool stdcall;
```

PEGetNthParamInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the name and type of a specified stored procedure parameter.

```
BOOL CRPE_API PEGetNthParamInfo (
    short printJob,
    short paramN,
    PEParameterInfo FAR *paramInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to gather information about a stored procedure parameter.
paramN	Specifies the number of the stored procedure parameter about which you want to gather information.
paramInfo	Specifies a pointer to <i>PEParameterInfo</i> , <i>Page 432</i> , that will hold the retrieved information.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetNthParamInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthParamInfo Lib "crpe.dll" (ByVal
printJob As Integer, ByVal ParamN As Integer, paramInfo As
PEParameterInfo) As Integer
```

32-bit

```
Declare Function PEGetNthParamInfo Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal paramN As Integer, paramInfo As
PEParameterInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthParamInfo (
    printJob: integer;
    paramN:     integer;
    var pInfo: PParameterInfo
) : Bool;
```

32-bit

```
function PEGetNthParamInfo (
    printJob: Word;
    paramN: Smallint;
    var pInfo: PParameterInfo
) : Bool stdcall;
```

PEGetNthSortField

Returns information about one of the sort fields in the specified report: that is, it returns the name of the field and the direction (ascending or descending) of the sort. The name of the sort field is returned as a string handle. This function is typically used as one of a series of functions: *PEGetNSortFields*, *Page 187* (called once), *PEGetNthSortField*, *PEGetHandleString*, *Page 170* (called together as many times as needed to identify the correct sort field), and *PESetNthSortField*, *Page 316* (called once when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then change an existing sort field and/or sort order at print time in response to a user selection.

```
BOOL CRPE_API PEGetNthSortField (
    short printJob,
    short sortFieldN,
    HANDLE FAR *nameHandle,
    short FAR *nameLength,
    short FAR *direction
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to retrieve sort field information.
sortFieldN	Specifies the number of the sort field you want to retrieve. The first sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1.

<i>Parameter</i>	<i>Description</i>
nameHandle	Specifies the pointer to the handle of the string containing the sort field name.
nameLength	Specifies the pointer to the length of the field name string (in bytes) including the terminating byte.
direction	Specifies a pointer to the sort direction. See <i>PESetNthSortField</i> , Page 316, for a list of sort direction constants.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Report | Record Sort Order

PEGetNthSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthSortField Lib "crpe.dll" (ByVal
printJob As Integer, ByVal SortNumber As Integer, NameHandle As
Integer, NameLength As Integer, Direction As Integer) As Integer
```

32-bit

```
Declare Function PEGetNthSortField Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal SortNumber As Integer, NameHandle As
Long, NameLength As Integer, Direction As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthSortField (
    printJob,
    sortFieldN: integer;
    var nameHandle: HWnd;
    var nameLength: integer;
    var direction: integer
  ): Boolean;
```

32-bit

```
function PEGetNthSortField (
    printJob: Word;
    sortFieldN: integer;
    var nameHandle: HWnd;
    var nameLength: Word;
    var direction: Word
): Bool stdcall;
```

PEGetNthSubreportInSection



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves a handle that is required to retrieve the name of the subreport.

```
DWORD CRPE_API PEGetNthSubreportInSection (
    short printJob,
    short sectionCode,
    short subreportN
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the primary report.
sectionCode	Specifies the code for the report section that contains the subreport. See <i>Working with section codes, Page 116</i> .
subreportN	Specifies the number of the subreport in the specified section. subreportN is zero based. The first report in the section will be 0, the second will be 1, etc. If there are no subreports in the section, the function will return 0.

Remarks

Use *PEGetSubreportInfo, Page 228*, to retrieve information about the subreport by passing the subreport handle returned by PEGetNthSubreportInSection.

PEGetNthSubreportInSection (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthSubreportInSection Lib "crpe.dll"  
    (ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal  
    subreportN As Integer) As Long
```

32-bit

```
Declare Function PEGetNthSubreportInSection Lib "crpe32.dll"  
    (ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal  
    subreportN As Integer) As Long
```

Delphi Syntax

16-bit

```
function PEGetNthSubreportInSection (  
    printJob: integer;  
    sectionCode: integer;  
    subreportN: integer  
): Longint;
```

32-bit

```
function PEGetNthSubreportInSection (  
    printJob: Word;  
    sectionCode: Smallint  
    subreportN: Smallint  
): DWORD stdcall;
```

PEGetNthTableLocation

Determines the location of a selected table used in the specified print job. This function is typically combined with *PESetNthTableLocation*, Page 319, to identify the location of a table and then to change it.

```
BOOL CRPE_API PEGetNthTableLocation (  
    short printJob,  
    short tableN,  
    PETableLocation FAR *location  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve information about a table's location.
tableN	Specifies the number of the table for which you want to retrieve location information. The first table is table 0.
location	Specifies the pointer to <i>PETableLocation</i> , <i>Page 457</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Database | Set Location

PEGetNthTableLocation (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPEGeNthTableLocation in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetNthTableLocation (
  printJob,
  tableN: integer;
  var location: PETableLocation
): Bool;
```

32-bit

```
function PEGetNthTableLocation(
  printJob: Word;
  tableN: integer;
  var location: PETableLocation
): Bool stdcall;
```

PEGetNthTableLogOnInfo

Retrieves log on information required by a report.

```
BOOL CRPE_API PEGetNthTableLogOnInfo (
    short printJob,
    short tableN,
    PELogOnInfo FAR *logOnInfo
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to get table log on information.
tableN	Specifies the number of the table from which you want to get log on information. The first table is table 0.
logOnInfo	Specifies the pointer to <i>PELogOnInfo</i> , Page 425 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function must be called after *PEOpenPrintJob*, [Page 248](#) (so you have the job handle), and before *PESetPrintJob*, [Page 348](#) (which needs the password set to print the report).
- password in PELogOnInfo will always be an empty string.

PEGetNthTableLogOnInfo (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPEGeNthTableLogOnInfo in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetNthTableLogOnInfo (
    printJob: integer;
    tableN: integer;
    var logOnInfo: PELogOnInfo
): Bool;
```

32-bit

```
function PEGetNthTableLogOnInfo (
    printJob: Word;
    tableN: integer;
    var logOnInfo: PELogOnInfo
): Bool stdcall;
```

PEGetNthTableSessionInfo

Gets the session information for a Microsoft Access table being used in your report. Many MS Access database tables require that a session be opened before the information in the table can be used. Use PEGetNthTableSessionInfo to obtain the session information (User ID, Password, and Session Handle) for a particular table.

```
BOOL CRPE_API PEGetNthTableSessionInfo (
    short printJob,
    short tableN,
    PESessionInfo FAR *sessionInfo
);
```

Parameters

Parameter	Description
printJob	Identifies the handle of the print job that uses the table for which you need to obtain the session information.
tableN	0 indexed table number indicating which table in the report you wish to obtain the session information for.
sessionInfo	Refer to <i>PESessionInfo</i> , Page 448, for further information.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function is only applicable for MS Access databases which require a session to be opened before the database is accessed.
- password in PESessionInfo will always be an empty string.

PEGetNthTableSessionInfo (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPEGeNthTableSessionInfo in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetNthTableSessionInfo (
    printJob: integer;
    tableN: integer;
    var sessionInfo: PESessionInfo
): Bool;
```

32-bit

```
function PEGetNthTableSessionInfo (
    printJob: Word;
    tableN: Integer;
    var sessionInfo: PESessionInfo
): Bool stdcall;
```

PEGetNthTableType

Allows the application to determine the type of each table.

This function is one of a series of functions that enable you to retrieve and update database information in an opened report so that the report can be printed using different server, database, user, and/or table location settings.

```
BOOL CRPE_API PEGetNthTableType (
    short printJob,
    short tableN,
    PETableType FAR *tableType
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to determine a table type.
tableN	Specifies the number of the table for which you want to determine the type. Table numbers start at 0. For example, if PEGetNTables returns 2, call PEGetNthTableType twice with table numbers of 0 and 1.
tableType	Specifies the pointer to <i>PETableType</i> , Page 459 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if anything goes wrong.

Remarks

The application can test DBType in *PETableType*, [Page 459](#), or test the database DLL name used to create the report. DBType is the structure returned by PEGetNthTableType. DLL names have the following naming convention:

- PDB*.DLL for standard (non-SQL) databases,
- PDS*.DLL for SQL/ODBC databases.

NOTE: While there are some differences in naming conventions between 16-bit and 32-bit versions, you can always use the 16-bit convention (above) and be confident that it will work.

- In the case of ODBC (PDSODBC.DLL) the DescriptiveName includes the ODBC data source name.
- All strings must be null-terminated.

PEGetNthTableType (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetNthTableType Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal TableN As Integer, TableType As  
PETableType) As Integer
```

32-bit

```
Declare Function PEGetNthTableType Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal TableN As Integer, TableType As  
PETableType) As Integer
```

Delphi Syntax

16-bit

```
function PEGetNthTableType (  
    printJob,  
    tableN: Integer;  
    var tableType: PETableType  
): Bool;
```

32-bit

```
function PEGetNthTableType (  
    printJob: Word;  
    tableN: Integer;  
    var tableType: PETableType  
): Bool stdcall;
```

PEGetPrintDate

Determines the print date (if any) that was specified with the report. Use this function to fetch the print date and pass back using *PESetPrintDate*, *Page 324*.

```
BOOL CRPE_API PEGetPrintDate (  
    short printJob,  
    short FAR *year,  
    short FAR *month,  
    short FAR *day  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to determine its print date setting.
year	Specifies a pointer to the year component of the print date.
month	Specifies a pointer to the month component of the print date.
day	Specifies a pointer to the day component of the print date.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

You change the print date, typically, when you want to run the report today yet have it appear to have been run on a different date. An example would be, if you were out of town on the last day of the previous month and you later want to run a report for that month and make it appear as if it were run on the last day of the month.

Related Menu Commands

Report | Set Print Date

PEGetPrintDate (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetPrintDate Lib "crpe.dll" (ByVal printJob  
As Integer, Date_Year As Integer, Date_Month As Integer,  
Date_Day As Integer) As Integer
```

32-bit

```
Declare Function PEGetPrintDate Lib "crpe32.dll" (ByVal printJob  
As Integer, Date_Year As Integer, Date_Month As Integer,  
Date_Day As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetPrintDate (   
    printJob: integer;  
    var year: integer;  
    var month: integer;  
    var day: integer  
  ): Bool;
```

32-bit

```
function PEGetPrintDate (
    printJob: Word;
    var year: Word;
    var month: Word;
    var day: Word
): Bool stdcall;
```

PEGetPrintOptions



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the print options specified for the report (the options that are set in the Print common dialog box) and uses them to fill in *PEPrintOptions*, *Page 435*. Use this function to fetch print options from the report in order to update them and pass back using *PESetPrintOptions*, *Page 326*.

```
BOOL CRPE_API PEGetPrintOptions (
    short printJob,
    PEPrintOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to find out which print options have been set using the Print common dialog box.
options	Specifies the pointer to <i>PEPrintOptions</i> , <i>Page 435</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

File | Print | Printer

PEGetPrintOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetPrintOptions Lib "crpe.dll" (ByVal  
printJob As Integer, Options As PEPrintOptions) As Integer
```

32-bit

```
Declare Function PEGetPrintOptions Lib "crpe32.dll" (ByVal  
printJob As Integer, Options As PEPrintOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEGetPrintOptions (  
    printJob: integer;  
    var options: PEPrintOptions  
): Bool;
```

32-bit

```
function PEGetPrintOptions (  
    printJob: Word;  
    var options: PEPrintOptions  
): Bool stdcall;
```

PEGetReportTitle



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Returns the handle of the report title string in the report summary information. If the job is a subreport, it returns the handle of the subreport name. This function is typically used as one of a series of functions (PEGetReportTitle, *PEGetHandleString*, *Page 170*, or *PESetReportTitle*, *Page 329*). The series can be used in a Custom-Print link to identify and then change an existing report title at print time in response to a user selection.

```

BOOL CRPE_API PEGetReportTitle (
    short printJob,
    HANDLE FAR *titleHandle,
    short FAR *titleLength
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job for which you want to get the report title.
titleHandle	Specifies a pointer to the handle of the title string.
titleLength	Specifies a pointer to the length of the title string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Report | Report Title

PEGetReportTitle (Other Declares)

VB Syntax

16-bit

```

Declare Function PEGetReportTitle Lib "crpe.dll" (ByVal printJob
As Integer, TitleHandle As Integer, TitleLength As Integer) As
Integer

```

32-bit

```

Declare Function PEGetReportTitle Lib "crpe32.dll" (ByVal
printJob As Integer, TitleHandle As Long, TitleLength As
Integer) As Integer

```

Delphi Syntax

16-bit

```
function PEGetReportTitle (
    printJob: integer;
    var titleHandle: HWnd;
    var titleLength: integer
): Bool;
```

32-bit

```
function PEGetReportTitle (
    printJob: Word;
    var titleHandle: HWnd;
    var titleLength: Word
): Bool stdcall;
```

PEGetReportSummaryInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Obtains summary information about the report such as the report title, author, and comments.

```
BOOL CRPE_API PEGetSummaryInfo (
    short printJob,
    PEReportSummaryInfo FAR *summaryInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to get the report summary information.
summaryInfo	Specifies a pointer to <i>PEReportSummaryInfo</i> , Page 440.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetReportSummaryInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetReportSummaryInfo Lib "crpe.dll" (ByVal  
printJob As Integer, summaryInfo As PEReportSummaryInfo) As  
Integer
```

32-bit

```
Declare Function PEGetReportSummaryInfo Lib "crpe32.dll" (ByVal  
printJob As Integer, summaryInfo As PEReportSummaryInfo) As  
Integer
```

Delphi Syntax

16-bit

```
function PEGetReportSummaryInfo ( //  
    printJob: integer;  
    var summaryInfo: PEReportSummaryInfo  
): Bool;
```

32-bit

```
function PEGetReportSummaryInfo ( //  
    printJob: integer;  
    var summaryInfo: PEReportSummaryInfo  
): Bool stdcall;
```

PEGetSectionCode

Retrieves the section code for the specified section. A section code indicates the section type (Page Header, Details, and so forth). If there are multiple group sections it also identifies the group number, and if there are multiple sections in an area it identifies the section number.

```
short CRPE_API PEGetSectionCode ( //  
    short printJob,  
    short sectionN  
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to identify a section code.
sectionN	Specifies the number of the section for which you want the section code. See <i>Working with section codes, Page 116</i> .

Return value

The section code for the specified section. See *Working with section codes, Page 116*.

PEGetSectionCode (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSectionCode Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal sectionN As Integer) As Integer
```

32-bit

```
Declare Function PEGetSectionCode Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sectionN As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetSectionCode (   
    printJob: integer;  
    sectionN: integer  
): integer;
```

32-bit

```
function PEGetSectionCode (   
    printJob: Word;  
    sectionN: Smallint  
): Smallint stdcall;
```

PEGetSectionFormat



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Retrieves the section format settings for selected sections in the specified report and supplies them as member values for *PESectionOptions*, *Page 444*. Use this function in order to update the section formats and pass them back using *PESetSectionFormat*, *Page 331*.

```
BOOL CRPE_API PEGetSectionFormat (
    short printJob,
    short sectionCode,
    PESectionOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to find what section options have been set for the report using the Format Section dialog box.
sectionCode	Specifies the code for the report section(s) for which you want to get section format information. See <i>Working with section codes</i> , <i>Page 116</i> .
options	Specifies the pointer to <i>PESectionOptions</i> , <i>Page 444</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Format | Section

PEGetSectionFormat (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSectionFormat Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, Options As  
PESectionOptions) As Integer
```

32-bit

```
Declare Function PEGetSectionFormat Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, Options As  
PESectionOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEGetSectionFormat (  
    printJob: integer;  
    sectionCode: integer;  
    var options: PESectionOptions  
): Bool;
```

32-bit

```
function PEGetSectionFormat (  
    printJob: Word;  
    sectionCode: integer;  
    var options: PESectionOptions  
): Bool stdcall;
```

PEGetSectionFormatFormula

Retrieves the current format formula for the specified section of the report.

```
BOOL CRPE_API PEGetSectionFormatFormula (
    short printJob,
    short sectionCode,
    short formulaName,
    HANDLE FAR *textHandle,
    short FAR *textLength
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job you want to query to find what section options have been set for the report using the Format Section dialog box. For more information, search for <i>Format Section dialog box</i> in Seagate Crystal Reports online Help.
sectionCode	Specifies the code for the report section(s) for which you want to get section format information. See the table of section constants supplied in <i>Working with section codes, Page 116</i> .
formulaName	Specifies the name of the section format formula.

Constant	Value	Meaning
PE_FFN_SECTION _VISIBILITY	58	Specifies that the section be visible. If not visible, no drill down will be available.
PE_FFN_SHOW_AREA	59	Specifies that the area be visible. If not visible, drill down will be available.
PE_FFN_NEW_PAGE _BEFORE	60	Specifies that a new page will be printed before the section.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FFN_NEW_PAGE _AFTER	61	Specifies that a new page will be printed after the section.
PE_FFN_KEEP_TOGETHER	62	Specifies that the section will be kept together.
PE_FFN_SUPPRESS_BLANK _SECTION	63	Specifies that if the section is blank, it will not be printed.
PE_FFN_RESET_PAGE_N _AFTER	64	Specifies that page numbering will reset after the section.
PE_FFN_PRINT_AT _BOTTOM_OF_PAGE	65	Specifies that the section will be printed at the bottom of the page.
PE_FFN_UNDERLAY _SECTION	66	Specifies that the section will overlay any sections following.
PE_FFN_SECTION_BACK _COLOUR	67	Specifies the background color of the section.

<i>Parameter</i>	<i>Description</i>
textHandle	A handle to the memory location of the text of the actual formula.
textLength	The length of the text string specified by the textHandle parameter. Use this value to allocate a buffer for the text.

Return Value

- TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- Not all formula names apply to all sections.

- Use the value returned by textLength to allocate memory for a buffer. Use PEGetHandleString to fill the buffer with the actual text of the formula.

PEGetSectionFormatFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSectionFormatFormula Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, ByVal  
formulaName As Integer, textHandle As Integer, textLength As  
Integer) As Integer
```

32-bit

```
Declare Function PEGetSectionFormatFormula Lib "crpe32.dll"  
(ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal  
formulaName As Integer, textHandle As Long, textLength As  
Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetSectionFormatFormula (  
  printJob: Word;  
  sectionCode: Word;  
  formulaName: Word;  
  var textHandle: HWnd;  
  var textLength: Word  
): Bool;
```

32-bit

```
function PEGetSectionFormatFormula (  
  printJob: Word;  
  sectionCode: Word;  
  formulaName: Word;  
  var textHandle: HWnd;  
  var textLength: Word  
): Bool stdcall;
```

PEGetSelectedPrinter

This function obtains information about the printer currently selected for the report. If a printer has been specified in Seagate Crystal Reports using the File | Printer Setup | Specific printer option, this call will return information about that printer. If the File | Printer Setup | Default printer option has been selected for the report, and custom options for the Default printer have been specified (Default Properties is toggled off in the Print Setup dialog box), information about the default printer specified under Windows Control Panel | Printers will be returned. If Default Properties is toggled on for the Default printer, this function will return a successful result, but the string handles will point to NULL strings.

16-bit

```
BOOL CRPE_API PEGetSelectedPrinter (
    short printJob,
    HANDLE FAR *driverHandle,
    short FAR *driverLength,
    HANDLE FAR *printerHandle,
    short FAR *printerLength,
    HANDLE FAR *portHandle,
    short FAR *portLength,
    DEVMODE FAR *FAR *mode
);
```

32-bit

```
BOOL CRPE_API PEGetSelectedPrinter (
    short printJob,
    HANDLE FAR *driverHandle,
    short FAR *driverLength,
    HANDLE FAR *printerHandle,
    short FAR *printerLength,
    HANDLE FAR *portHandle,
    short FAR *portLength,
    DEVMODEA FAR *FAR *mode
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job that you want to query to get information on the non-default printer that has been selected with the report.

<i>Parameter</i>	<i>Description</i>
driverHandle	Specifies a pointer to the handle of the printer driver for the printer that is selected with the print job.
driverLength	Specifies a pointer to the length of the printer driver name.
printerHandle	Specifies a pointer to the handle of the printer that is selected with the print job.
printerLength	Specifies a pointer to the length of the printer name.
portHandle	Specifies a pointer to the handle of the port to which the selected printer is connected.
portLength	Specifies a pointer to the length of the port name.
mode	Specifies a pointer to the Windows API structure <i>DEVMODE</i> , Page 353 (Use <i>DEVMODEA</i> for 32-bit.) If <i>UNICODE</i> is defined in your application, this will be a <i>DEVMODEA</i> structure.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

Use *PEGetHandleString*, [Page 170](#), to obtain the actual strings pointed to by the string handles returned.

Related Menu Commands

File | Printer Setup

PEGetSelectedPrinter (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for *crPEGetSelectedPrinter* in Developer's online Help.

Delphi Syntax

16-bit

```
function PEGetSelectedPrinter (
    printJob: integer;
    var driverHandle: HWnd;
    var driverLength: integer;
    var printerHandle: HWnd;
    var printerLength: integer;
    var portHandle: HWnd;
    var portLength: integer;
    var mode: PDevMode
): Bool;
```

32-bit

```
function PEGetSelectedPrinter (
    printJob: Word;
    var driverHandle: Hwnd;
    var driverLength: Word;
    var printerHandle: Hwnd;
    var printerLength: Word;
    var portHandle: Hwnd;
    var portLength: Word;
    var mode: PDeviceModeA
): Bool stdcall;
```

PEGetSelectionFormula

Returns the string handle for the selection formula used in the specified report. This function is typically used as one of a series of functions (*PEGetSelectionFormula*, *PEGetHandleString*, *Page 170*, or *PESetSelectionFormula*, *Page 336*). The series can be used in a Custom-Print link to identify and then change an existing record selection formula at print time in response to a user selection.

```
BOOL CRPE_API PEGetSelectionFormula (
    short printJob,
    HANDLE FAR *textHandle,
    short FAR *textLength
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to retrieve the selection formula string.
textHandle	Specifies the pointer to the handle of the string containing the formula text.
textLength	Specifies the pointer to the length of the formula string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Report | Edit Record Selection Formula

PEGetSelectionFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSelectionFormula Lib "crpe.dll" (ByVal  
printJob As Integer, TextHandle As Integer, TextLength As  
Integer) As Integer
```

32-bit

```
Declare Function PEGetSelectionFormula Lib "crpe32.dll" (ByVal  
printJob As Integer, TextHandle As Long, TextLength As Integer)  
As Integer
```

Delphi Syntax

16-bit

```
function PEGetSelectionFormula ( //  
  printJob: integer;  
  var textHandle: HWnd;  
  var textLength: integer  
 ): Boolean;
```

32-bit

```
function PEGetSelectionFormula (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
): Bool stdcall;
```

PEGetSQLQuery

Returns the same query that appears in the Show SQL Query dialog box in Seagate Crystal Reports, in a syntax that's specific to the database driver you're using.

You can use this function to retrieve the SQL query that will be generated to print the report, and you can update the query using *PESetSQLQuery*, *Page 337*.

```
BOOL CRPE_API PEGetSQLQuery (
    short printJob,
    HANDLE FAR *textHandle,
    short FAR *textLength
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to retrieve the SQL query.
textHandle	Specifies the pointer to the handle of the string containing the SQL query string.
textLength	Specifies the pointer to the length of the SQL query string (in bytes) including the terminating byte.

Return Value

TRUE (1) if the call is successful; (0) if an error occurs.

Related Menu Commands

Database | Show SQL Query

PEGetSQLQuery (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSQLQuery Lib "crpe.dll" (ByVal printJob As Integer, TextHandle As Integer, TextLength As Integer) As Integer
```

32-bit

```
Declare Function PEGetSQLQuery Lib "crpe32.dll" (ByVal printJob As Integer, TextHandle As Long, TextLength As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetSQLQuery (
    printJob: integer;
    var textHandle: HWnd;
    var textLength: integer
): Bool;
```

32-bit

```
function PEGetSQLQuery (
    printJob: Word;
    var textHandle: HWnd;
    var textLength: Word
): Bool stdcall;
```

PEGetSubreportInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

.Retrieves information about the specified subreport.

```
BOOL CRPE_API PEGetSubreportInfo (
    short printJob,
    DWORD subreportHandle,
    PESubreportInfo FAR *subreportInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the primary report that contains the subreport about which you want to retrieve information.
subreportHandle	Specifies the handle of the subreport about which you want to retrieve information.
subreportInfo	Specifies a pointer to <i>PESubreportInfo</i> , Page 455, that will be used for holding the information once it is retrieved.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetSubreportInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetSubreportInfo Lib "crpe.dll" (ByVal
    printJob As Integer, ByVal subreportHandle As Long,
    subreportInfo As PESubreportInfo) As Integer
```

32-bit

```
Declare Function PEGetSubreportInfo Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal subreportHandle As Long,  
subreportInfo As PESubreportInfo) As Integer
```

Delphi Syntax

16-bit

```
function PEGetSubreportInfo (  
    printJob: integer;  
    subreportHandle: Longint;  
    var subreportInfo: PESubreportInfo  
): Bool;
```

32-bit

```
function PEGetSubreportInfo (  
    printJob: Word;  
    subreportHandle: DWORD;  
    var subreportInfo: PESubreportInfo  
): Bool stdcall;
```

PEGetTrackCursorInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Track cursors can be specified for different report areas and report objects in the preview window. This function get track cursor information for a specified job.

```
BOOL CRPE_API PEGetTrackCursorInfo (  
    short printJob,  
    PETrackCursorInfo FAR *cursorInfo  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you wish to get track cursor information for.

<i>Parameter</i>	<i>Description</i>
cursorInfo	Specifies a pointer to <i>PETrackCursorInfo</i> , <i>Page 461</i> , that will be used for getting track cursor information.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEGetTrackCursorInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetTrackCursorInfo Lib "crpe.dll" (ByVal
printJob As Integer, cursorInfo As PETrackCursorInfo) As
Integer;
```

32-bit

```
Declare Function PEGetTrackCursorInfo Lib "crpe.dll" (ByVal
printJob As Integer, cursorInfo As PETrackCursorInfo) As
Integer;
```

Delphi Syntax

16-bit

```
function PEGetTrackCursorInfo (
  printJob: Integer;
  var cursorInfo: PETrackCursorInfo
): BOOL;
```

32-bit

```
function PEGetTrackCursorInfo (
  printJob: smallint;
  var cursorInfo: PETrackCursorInfo
): Bool stdcall;
```

PEGetVersion

Returns the version number of the DLL or the Crystal Report Engine. The high-order byte is the major version number and the low-order byte is the minor version number. This function can be used whenever you build functionality into a report that may not be available in earlier versions of the Crystal Report Engine and you need to verify the version number first. The function can be a handy safeguard for users who have more than one version of the Crystal Report Engine with the older version earlier in the path than the new version.

```
short CRPE_API PEGetVersion (
    short versionRequested
);
```

Parameters

Parameter	Description		
versionRequested	Specifies whether the DLL or Crystal Report Engine version is being requested. Use one of the following values:		
Constant	Value	Meaning	
PE_GV_DLL	100	Returns the version of the DLL (CRPE/CRPE32).	
PE_GV_ENGINE	200	Returns the version of the Crystal Report Engine.	

Return Value

The version number of the DLL or the Crystal Report Engine.

PEGetVersion (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetVbVersion Lib "crpe.dll" (ByVal Version As Integer) As Integer
```

32-bit

```
Declare Function PEGetVbVersion Lib "crpe32.dll" (ByVal Version As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetVersion (
    versionRequested: integer
): integer;
```

32-bit

```
function PEGetVersion (
    versionRequested: integer
): Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEGetVersion (CWORD) CRPE.DLL
```

PEGetWindowHandle

Returns the handle of the preview window. This function can be used in a Custom-Print link if you want to do something with the preview window (move it, change its size, etc.).

PEGetWindowHandle can also be used to determine if the user has already closed the preview window.

```
HWND CRPE_API PEGetWindowHandle (
    short printJob
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve the preview window handle. If two or more preview windows are open, this function applies only to the most recently created preview window.

Return Value

This function returns the preview window handle if it is successful, 0 if an error occurs or if the preview window has already been closed.

Remarks

This function can be used after *PESetPrintJob*, *Page 348*, and then, only if you have created a preview window.

PEGetWindowHandle (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetWindowHandle Lib "crpe.dll" (ByVal  
printJob As Integer) As Integer
```

32-bit

```
Declare Function PEGetWindowHandle Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEGetWindowHandle (   
    printJob: integer  
 ): HWnd;
```

32-bit

```
function PEGetWindowHandle (   
    printJob: Word  
 ): HWnd stdcall;
```

dBASE for Windows Syntax

```
EXTERN CHANDLE PEGetWindowHandle (CWORD) CRPE.DLL
```

PEGetWindowOptions



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

You can configure the preview window look and functionality by calling this function. You can also determine whether the preview window has a group tree window, whether it can be drill down if there are hidden groups, and whether the close button, refresh button, and print setup button are shown. This function returns the current report preview window configuration.

```
BOOL CRPE_API PEGetWindowOptions (
    short printJob,
    PEWindowOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to retrieve the preview window handle. If two or more preview windows are open, this function applies only to the most recently created preview window.
PEWindowOptions	Specifies a pointer to <i>PEWindowOptions</i> , Page 470, that will be used for holding the information once it is retrieved.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

When hasGroupTree is True, it does not mean there will be a group tree in the preview window. The hasGroupTree option and the report option (Create Group Tree in Seagate Crystal Reports) together determine the group tree visibility in the preview window. Both options must be True for the group tree to be shown.

PEGetWindowOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PEGetWindowOptions Lib "crpe.dll" (ByVal  
printJob As Integer, Options As PEWindowOptions) As Integer
```

32-bit

```
Declare Function PEGetWindowOptions Lib "crpe32.dll" (ByVal  
printJob As Integer, Options As PEWindowOptions) As Integer
```

Delphi Syntax

16-bit

```
function PEGetWindowOptions (  
    printJob: Word;  
    var options: PEWindowOptions  
) : Bool;
```

32-bit

```
function PEGetWindowOptions (  
    printJob: Word;  
    var options: PEWindowOptions  
) : Bool stdcall;
```


PEHasSavedData

Queries a report to find if data is saved with it in memory. With this information, you can determine whether or not the data needs to be refreshed before the report is printed.

```
BOOL CRPE_API PEHasSavedData (
    short printJob,
    BOOL FAR *hasSavedData
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to find if it has saved data.
hasSavedData	Specifies a pointer to a memory address that indicates whether or not there is data saved with the report.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

A report may or may not have saved data when a print job is first opened from a report file. Since data is saved during a print, however, it will always have saved data immediately after a report is printed.

Use *PEDiscardSavedData*, *Page 140*, to release the saved data associated with a report. The next time the report is printed, it will get current data from the database.

The default behavior is for a report to use its saved data rather than refresh its data from the database when printing a report.

Related Menu Commands

File | Save Data with Report

PEHasSavedData (Other Declares)

VB Syntax

16-bit

```
Declare Function PEHasSavedData Lib "crpe.dll" (ByVal printJob  
As Integer, HasSavedData As Integer) As Integer
```

32-bit

```
Declare Function PEHasSavedData Lib "crpe32.dll" (ByVal printJob  
As Integer, HasSavedData As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEHasSavedData (  
    printJob: integer;  
    var hasSavedData: Bool  
): Bool;
```

32-bit

```
function PEHasSavedData(  
    printJob: Word;  
    var hasSavedData: Bool  
): Bool stdcall;
```

PEIsPrintJobFinished

Monitors the print job to see if it is finished or still in progress. You can use this function any time you have a call that is contingent on a print job being finished.

```
BOOL CRPE_API PEIsPrintJobFinished (  
    short printJob  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you are querying to find if it has finished printing.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

PEIsPrintJobFinished will return TRUE immediately after the report has been displayed in the preview window, even if that preview window is still open.

PEIsPrintJobFinished (Other Declares)

VB Syntax

16-bit

```
Declare Function PEIsPrintJobFinished Lib "crpe.dll" (ByVal  
printJob As Integer) As Integer
```

32-bit

```
Declare Function PEIsPrintJobFinished Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEIsPrintJobFinished (  
    printJob: integer  
): Bool;
```

32-bit

```
function PEIsPrintJobFinished (  
    printJob: Word  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEIsPrintJobFinished (CWORD) CRPE.DLL
```

PELogOffServer

Logs off the specified server. Use any time you have to log off a specified server.

```
BOOL CRPE_API PELogOffServer (
    char FAR *dllName,
    PELogOnInfo FAR *logOnInfo
);
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL for the datasource you want to log off, for example, “PDSODBC.DLL”. Note that the dllName must be enclosed in quotes. DLL names have the following naming convention: PDB*.DLL for standard (non-SQL) databases, PDS*.DLL for SQL/ODBC databases (16-bit and 32-bit).
logOnInfo	Specifies the pointer to <i>PELogOnInfo</i> , Page 425 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- *PELogOnServer*, [Page 241](#), and *PELogOffServer* can be called at any time to log on and off a database server. These functions are not required if the function *PESetNthTableLogOnInfo*, [Page 320](#), was already used to log on to a table.
- This function requires a database DLL name which can be retrieved using *PEGetNthTableType*, [Page 207](#).

Related Menu Commands

Database | Log Off Server

PELogOffServer (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPELogoffServer in Developer's online Help.

Delphi Syntax

16-bit

```
function PELogOffServer (
    dllName:PChar;
    var logOnInfo: PELogOnInfo
): Bool;
```

32-bit

```
function PELogOffServer(
    dllName: PChar;
    var logOnInfo: PELogOnInfo
): Bool stdcall;
```

PELogOnServer

Logs on to the specified server.

```
BOOL CRPE_API PELogOnServer (
    char FAR *dllName,
    PELogOnInfo FAR *logOnInfo
);
```

Parameters

<i>Parameter</i>	<i>Description</i>
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to, for example, "PDSODBC.DLL". Note that the dllName must be enclosed in quotes. DLL names have the following naming convention: PDB*.DLL for standard (non-SQL) databases, PDS*.DLL for SQL/ODBC databases.
logOnInfo	Specifies the pointer to <i>PELogOnInfo</i> , <i>Page 425</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- PELogOnServer and *PELogOffServer*, *Page 240*, can be called to log on and off a database server. These functions are not required if the function *PESetNthTableLogOnInfo*, *Page 320*, was already used to set the password for a table.
- This function requires a database DLL name, which can be retrieved using *PEGetNthTableType*, *Page 207*.
- This function can also be used for non-SQL tables, such as password-protected Paradox tables. Call this function to set the password for the Paradox DLL before beginning printing.
- When printing using *PStartPrintJob*, *Page 348*, the ServerName passed in PELogOnServer must agree exactly with the server name stored in the report. If this is not true, use *PESetNthTableLogOnInfo*, *Page 320*, to perform logging on instead.
- The following points need to be considered when deciding whether to use PELogOnServer or PESetNthTableLogOnInfo:
 - PELogOnServer is easier to call than PESetNthTableLogOnInfo and it can be called at any time. You must know the database DLL name to make this call, however.
 - PESetNthTableLogOnInfo is more flexible than PELogOnServer. It allows you to override any of the log on parameters. This call must be called after *PEOpenPrintJob*, *Page 248*.

Remarks

The 16-bit file name is needed for both 32-bit and 16-bit.

Related Menu Commands

Database | Log On Server

PELogOnServer (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPELogOnServer in Developer's online Help.

Delphi Syntax

16-bit

```
function PELogOnServer (
    dllName: PChar;
    var logOnInfo: PELogOnInfo
): Bool;
```

32-bit

```
function PELogOnServer(
    dllName: PChar;
    var logOnInfo: PELogOnInfo
): Bool stdcall;
```

PELogOnSQLServerWithPrivateInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Enables the Crystal Report Engine to “piggyback” your application’s existing connection to a Server. If you are already logged on, this function lowers the number of connections established by a workstation, thus reducing application time and network traffic. It also prevents a Seagate Crystal Reports Log Off call from disconnecting an application’s existing connection to the Server.

```
BOOL CRPE_API PELogOnSQLServerWithPrivateInfo (
    char FAR *dllName,
    void FAR *privateInfo
);
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL that was used in establishing the connection to the server when the report was first created. Thus, if a report was created using an ODBC datasource, the Seagate Crystal Reports DLL is PDSODBC.DLL.

<i>Parameter</i>	<i>Description</i>
privateInfo	In the application, a connection to the server has to have been established and this in turn generates a Handle to a Database Connection (HDBC). This parameter specifies the application's handle to the connection. This makes Seagate Crystal Reports aware of the existing connection so it can use it instead of establishing a new one. Since the reports with which this function works are based on ODBC, this parameter is actually an ODBC HDBC.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function can only be used with database connections established by ODBC or Q+E Library 2.0. Any other database connections can not be accessed by this function.

If the application uses the Q+E Library to connect, get the ODBC HDBC by using the following function calls:

```
qeConnect (opens a connection to the server)
qeGetODBCHdbc (returns the ODBC hdbc)
```

NOTE: See the Intersolv DataDirect Developer's Toolkit for more information.

If the application uses ODBC to connect, get the ODBC HDBC by using the following function calls:

SQLAllocEnv

«Initializes the ODBC call level interface and allocates memory for an environment handle.»

SQLAllocConnect

«Returns an ODBC HDBC.»

NOTE: See the ODBC documentation for more information.

PELogOnSQLServerWithPrivateInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PELogOnSQLServerWithPrivateInfo Lib "crpe.dll"  
    (ByVal DLLName As String, ByVal PrivateInfo As Long) As Integer
```

32-bit

```
Declare Function PELogOnSQLServerWithPrivateInfo Lib  
    "crpe32.dll" (ByVal DLLName As String, ByVal PrivateInfo As  
    Long) As Integer
```

Delphi Syntax

16-bit

```
function PELogOnSQLServerWithPrivateInfo (  
    dllName: PChar;  
    privateInfo: Pointer  
): Bool;
```

32-bit

```
function PELogOnSQLServerWithPrivateInfo (  
    dllName: PChar;  
    privateInfo: Pointer  
): Bool stdcall;
```

PENextPrintWindowMagnification

Changes the preview window magnification to the next magnification level in order. Use this function to cycle through the three levels of preview window magnification whenever the report has been printed to a preview window. The three levels of magnification are: Full Page, Fit One Side, and Fit Both Sides.

```
BOOL CRPE_API PENextPrintWindowMagnification (  
    short printJob  
) ;
```

Parameters

Parameter	Description
printJob	The handle of the print job for the report appearing in the preview window.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function cycles to the next level of magnification according to the following order: Full Page, Fit One Side, Fit Both Sides, Full Page, etc. Each time the function is called, the preview window cycles to the next magnification level in order.

PENextPrintWindowMagnification (Other Declares)

VB Syntax

16-bit

```
Declare Function PENextPrintWindowMagnification Lib "crpe.dll"  
          (ByVal printJob As Integer) As Integer
```

32-bit

```
Declare Function PENextPrintWindowMagnification Lib "crpe32.dll"  
          (ByVal printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PENextPrintWindowMagnification (  
  printJob: integer  
): Bool;
```

32-bit

```
function PENextPrintWindowMagnification (  
  printJob: Word  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PENextPrintWindowMagnification (CWORD) CRPE.DLL
```

PEOpenEngine

Prepares the Crystal Report Engine for requests. This function is a necessary part of any Custom-Print link. It is also required for any Print-Only link in which you want the report to print to a window that is to remain visible after the report is printed. It is not necessary to use this function with a Print-Only link where you are directing the report to a printer.

```
BOOL CRPE_API PEOpenEngine ();
```

Return Value

TRUE (1) if Action is satisfactory; FALSE (0) if the call fails.

Remarks

This function must be called before any other Crystal Report Engine function. If an error occurs in the PEOpenEngine function call, *PEGetErrorCode*, *Page 152*, can be passed a print job value of zero to obtain error information.

PEOpenEngine (Other Declares)

VB Syntax

16-bit

```
Declare Function PEOpenEngine Lib "crpe.dll" () As Integer
```

32-bit

```
Declare Function PEOpenEngine Lib "crpe32.dll" () As Integer
```

Delphi Syntax

16-bit

```
function PEOpenEngine
): Bool;
```

32-bit

```
function PEOpenEngine  
    ) : Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEOpenEngine () CRPE.DLL
```

PEOpenPrintJob

Prepares to print a report and returns a number which identifies the particular print job. The number returned is called a print job handle, and must be used in all subsequent calls related to the new print job (where a print job handle is required). This function is used as a mandatory part of a Custom-Print link to generate the return of a print job handle. The handle is then used as the printJob parameter in each additional Custom-Print link function call (where a printJob parameter is required).

```
short PEOpenPrintJob (  
    char *reportFilePath  
) ;
```

Parameters

Parameter	Description
reportFilePath	Specifies the file name and path of the report you want to open. You must enclose this parameter in quotes.

Return Value

Returns the job number if no error occurs; returns 0 if the report file does not exist or if an error occurs.

Remarks

- This function must be called before most other Crystal Report Engine functions are used.
- Only one print job can be configured at a time.
- *PEClosePrintJob*, Page 129, must be called later to close the job.
- Report Path\Filename must be enclosed in quotes, for example:

```
PEOpenPrintJob  
( "C:\CRW\REPORT1.RPT" );
```

NOTE: In C or C++ the slashes in the string (\) must each be entered as a double slash (\).\.

This function opens the print job with the printer selected in the report (via the Print | Select Printer menu command) or the default printer (if no replacement printer has been selected in the report).

PEDOpenPrintJob (Other Declares)

VB Syntax

16-bit

```
Declare Function PEDOpenPrintJob Lib "crpe.dll" (ByVal RptName As String) As Integer
```

32-bit

```
Declare Function PEDOpenPrintJob Lib "crpe32.dll" (ByVal RptName As String) As Integer
```

Delphi Syntax

16-bit

```
function PEDOpenPrintJob (
    reportFilePath: PChar
): integer;
```

32-bit

```
function PEDOpenPrintJob (
    reportFilePath: PChar
): Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEDOpenPrintJob (CSTRING) CRPE.DLL
```

PEOpenSubreport



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Opens the named subreport and returns a number which identifies the subreport. The number returned is called a print job handle, and must be used in all subsequent calls related to the subreport (where a print job handle is required).

```
short CRPE_API PEOpenSubreport (
    short parentJob,
    char FAR *subreportName
);
```

Parameters

Parameter	Description
parentJob	Specifies the handle of the primary report (the report that contains the subreport). This is the handle returned from PEOpenPrintJob.
subreportName	Specifies a pointer to the name of the subreport you want to open. This is retrieved using PEGetSubreportInfo, Page 228.

Return Value

Returns the job number of the subreport if no error occurs; returns 0 if the subreport does not exist or if an error occurs.

Remarks

- This function must be called before any other Crystal Report Engine functions related to the subreport are used.
- PECloseSubreport must be called later to close the job.

PEOpenSubreport (Other Declares)

VB Syntax

16-bit

```
Declare Function PEOpenSubreport Lib "crpe.dll" (ByVal parentJob  
As Integer, ByVal subreportName As String) As Integer
```

32-bit

```
Declare Function PEOpenSubreport Lib "crpe32.dll" (ByVal  
parentJob As Integer, ByVal subreportName As String) As Integer
```

Delphi Syntax

16-bit

```
function PEOpenSubreport (  
  parentJob: integer;  
  subreportName: PChar  
): Integer;
```

32-bit

```
function PEOpenSubreport (  
  parentJob: Word;  
  subreportName: PChar  
): Word stdcall;
```

PEOutputToFile

Description

Prepares to direct printed output to a file. Data can be saved in a wide variety of formats, and, if you select character separated format, you can even specify:

- the string you want to use to separate the fields (at most 16 characters long), and
- your choice of any single character for enclosing alphanumeric field data.

NOTE: This structure does not appear in the .H file, .BAS files, .PAS. You must declare it yourself.

```

BOOL FAR PASCAL PEOutputToFile (
    short printJob,
    char FAR *outputFilePath,
    short type,
    void FAR *options
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	A handle to the current print job that is being sent to a disk file.
outputFilePath	The full path of the file that the report will be sent to.
type	A value indicating the format type of the file that the report is being sent to. The following table indicates possible values.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FT_RECORD	0	Saves data in record style format as column of values. Doesn't use commas or separators. Outputs every record with a fixed field width.
PE_FT_TABSEPARATED	1	Presents data in tabular form. Encloses alphanumeric field data in quotes and separates fields with tabs.
PE_FT_TEXT	2	Saves the data in ASCII text format with all values separated by spaces.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FT_DIF	3	Saves the data in DIF (data interchange format) format. This format is often used for the transfer of data between different spreadsheet programs.
PE_FT_CSV	4	Saves the data as comma separated values. Encloses alphanumeric field data in the character of your choice (typically single or double quotes) and separates fields with commas.
PE_FT_CHARSPEARATED	5	Saves the data as character separated values. Encloses alphanumeric field data in quotes and separates fields with the character of your choice.
PE_FT_TABFORMATTED	6	Saves the data in ASCII text format with all values separated by tabs.

<i>Parameter</i>	<i>Description</i>
options	A pointer to a structure containing additional information required for sending the report to a disk file. The structure used depends upon the format type specified in the type parameter. For all formats other than PE_FT_CHARSEPARATED, use <i>PEPrintFileOptions</i> , <i>Page 434</i> . If you specify PE_FT_CHARSEPARATED as your format, use <i>PECharSepFileOptions</i> , <i>Page 389</i> .

Return Value

TRUE (1) if the output can be sent to the file successfully, FALSE (0) if the output can not be sent to the file.

Remarks

This functions should be used only for backward compatibility with earlier versions of Seagate Crystal Reports. For new applications, use *PEExportTo*, *Page 146*.

PEOutputToPrinter

Prepares to direct printed output to a printer.

- If a printer has been specified via *PESelectPrinter*, *Page 267*, output will be sent to that printer.
- If there is no *PESelectPrinter* selection but there is a printer specified in the report via the Print | Select Printer menu command, output will be sent to that printer.
- If there is no *PESelectPrinter* selection, and there is no printer specified in the report, output will be to the Windows default printer.
- *PEOpenPrintJob*, *Page 248*, opens the print job with the printer specified in the report (if there is one) or with the Windows default printer (if no printer is specified in the report).

NOTE: The sequence of calls that follows may help to explain printer output concepts as well as potential problems. Assume that a printer is specified in the report via the Print|Select Printer menu command.

PEOpenPrintJob

```
// Opens the job with printer specified in report, or, if none
// is specified, the Windows default printer. The printer the
// job opens with is Printer #1.
```

PEOutputToWindow

```
// Directs the output to the preview window.
```

PEStartPrintJob

```
// Report is printed in the preview window based on Printer #1.
```

PEOutputToPrinter

```
// Directs output to the printer.
```

PESelectPrinter

```
// Specifies 2nd printer, Printer #2. This overrides Printer #1.
```

PEStartPrintJob

```
// Report is printed on Printer #2. Window output and printer  
// output are based on two different printers and may cause  
// confusion.
```

PEClosePrintJob

If one printer is set for landscape output, for example, and the other for portrait output, the sequence of calls above will print an entirely different report in the preview window than what actually appears on paper.

Make certain to sequence your function calls to get the output desired.

```
BOOL CRPE_API PEOutputToPrinter (  
    short printJob,  
    short nCopies  
) ;
```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job you want to send to a printer.
nCopies	Specifies the number of report copies you want printed.

Return Value

TRUE (1) if the output can be sent to the printer successfully; FALSE (0) if the output can not be sent to the printer.

Remarks

This function supersedes PEOOutputToDefaultPrinter which was available in earlier versions of the Crystal Report Engine.

PEOutputToPrinter (Other Declares)

VB Syntax

16-bit

```
Declare Function PEOOutputToPrinter Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal NCopies As Integer) As Integer
```

32-bit

```
Declare Function PEOOutputToPrinter Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal nCopies As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEOOutputToPrinter (   
    printJob: integer;  
    nCopies: integer  
): Bool;
```

32-bit

```
function PEOOutputToPrinter (   
    printJob: Word;  
    nCopies: integer  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEOOutputToPrinter (CWORD, CWORD) CRPE.DLL
```

POutputToWindow

Prepares to direct printed output to a preview window. This function is used as part of a Custom-Print link to whenever you want the report printed to the preview window instead of to the printer.

```
BOOL CRPE_API POutputToWindow (
    short printJob,
    char FAR *title,
    short left,
    short top,
    short width,
    short height,
    long style,
    HWND parentWindow
);
```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job you want to print in the preview window.
title	Specifies a pointer to the null-terminated string that contains the title you want to appear in the preview window title bar.
left	Specifies the x coordinate of the upper left corner of the preview window, in pixels.*
top	Specifies the y coordinate of the upper left corner of the preview window, in pixels.*
width	Specifies the width of the preview window, in pixels.
height	Specifies the height of the preview window, in pixels.

* For a top-level preview window, the top left corner is relative to the origin of the screen. For an MDI child preview window, the top left corner is relative to the origin of the frame window's client area. For a child preview window, the top left corner is relative to the origin of the parent window's client area.

<i>Parameter</i>	<i>Description</i>
style	Specifies the style of the window being created. Style settings can be combined using the bitwise "OR" operator. You can specify any of the following window styles:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
WS_MINIMIZE	536870912	Make a window of minimum size.
WS_VISIBLE	268435456	Make a window that is visible when it first appears (for overlapping and pop-up windows).
WS_DISABLED	134217728	Make a window that is disabled when it first appears.
WS_CLIPSIBLINGS	67108864	Clip child windows with respect to one another.
WS_CLIPCHILDREN	33554432	Exclude the area occupied by child windows when drawing inside the parent window.
WS_MAXIMIZE	16777216	Make a window of maximum size.
WS_CAPTION	12582912	Make a window that includes a title bar.
WS_BORDER	8388608	Make a window that includes a border.
WS_DLGFREAME	4194304	Make a window that has a double border but no title.
WS_VSCROLL	2097152	Make a window that includes a vertical scroll bar.
WS_HSCROLL	1048576	Make a window that includes a horizontal scroll bar.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
WS_SYSMENU	524288	Include the system menu box.
WS_THICKFRAME	262144	Include the thick frame that can be used to size the window.
WS_MINIMIZEBOX	131072	Include the minimize box.
WS_MAXIMIZEBOX	65536	Include the maximize box.
CW_USEDFAULT	-32768	Assign the child window the default horizontal and vertical position, and the default height and width.

<i>Parameter</i>	<i>Description</i>
parentWindow	Specifies the handle to the parentWindow if the preview window is a child of that window.

Return Value

TRUE (1) if the output can be sent to the window successfully; FALSE (0) if the output can not be sent to the window.

Remarks

- If *parentWindow* is null, the preview window is a top-level window (i.e., not a child of any other window). For a top-level preview window, the top left corner is relative to the origin of the screen. *Left* and *top* can be CW_USDEFAULT to put the window at a default location. *Width* and *height* can also be CW_USEDFAULT to give the window a default size.
- If the MDI frame window parent handle is specified, the report preview window will show up in the client area of the MDI parent. If the MDI frame window child window handle is specified (child window must be created) the report preview window will show up in the child window.
- If *parentWindow* is the handle of some other window, the preview window is a child of that window. For a child preview window, the top left corner is relative to the origin of the parent window's client area.

NOTE: Visual Basic developers can cut and paste a declaration for `CW_USEDEFAULT` into their application.

- For VB 4 (32-bit), cut the declaration from `c:\vb\winapi\win32api.txt`
- For VB 4 (16-bit), cut the declaration from `c:\vb\winapi\win31api.txt`
- For VB 3, cut the declaration from `c:\vb\winapi\win30api.txt`

If the preview window is a top-level window or an MDI child window and `style` is 0, a style of:

```
(WS_VISIBLE | WS_THICKFRAME | WS_SYSMENU | WS_MAXIMIZEBOX |
WS_MINIMIZEBOX)
```

is used instead.

- That is, the default window is a visible window with a thick frame that can be used for sizing the window. The window includes a system menu box, and a maximize and minimize box.
- The preview window is created when `PEStartPrintJob`, Page 348, is called.

PEOutputToWindow (Other Declares)

VB Syntax

16-bit

```
Declare Function PEOutputToWindow Lib "crpe.dll" (ByVal printJob
As Integer, ByVal Title As String, ByVal Left As Integer, ByVal
Top As Integer, ByVal Width As Integer, ByVal Height As Integer,
ByVal Style As Long, ByVal PWindow As Integer) As Integer
```

32-bit

```
Declare Function PEOutputToWindow Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal Title As String, ByVal Left As Long,
ByVal Top As Long, ByVal Width As Long, ByVal Height As Long,
ByVal style As Long, ByVal PWindow As Long) As Integer
```

Delphi Syntax

16-bit

```
function PEOutputToWindow (
    printJob: integer;
    title: PChar;
    left,
    top,
    width,
    height: integer;
    style: longint;
    parentWindow: HWnd
): Bool;
```

32-bit

```
function PEOutputToWindow (
    printJob: Word;
    title: PChar;
    left: longint;
    top: longint;
    width: longint;
    height: longint;
    style: longint;
    parentWindow: HWnd
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEOutputToWindow (CWORD, CSTRING, CWORD, CWORD, CWORD,
CWORD, CLONG, CHANDLE) CRPE.DLL
```

PEPrintControlsShowing



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Checks if the print controls are displayed in the preview window. Use this function to fetch the visible print controls and pass back using *PEShowPrintControls*, *Page 346*.

```
BOOL CRPE_API PEPrintControlsShowing (
    short printJob,
    BOOL FAR *controlsShowing
);
```

Parameters

Parameter	Description
printJob	Specifies the handle for the print job for which you want to check if the print controls will be displayed when the job is sent to a preview window.
controlsShowing	Returns a pointer to TRUE (1) if the print controls will be shown, FALSE (0) if they will be hidden.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEPrintControlsShowing (Other Declares)

VB Syntax

16-bit

```
Declare Function PEPrintControlsShowing Lib "crpe.dll" (ByVal  
printJob As Integer, ControlsShowing As Integer) As Integer
```

32-bit

```
Declare Function PEPrintControlsShowing Lib "crpe32.dll" (ByVal  
printJob As Integer, ControlsShowing As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEPrintControlsShowing (   
    printJob: integer;  
    var controlsShowing: Bool  
): Bool;
```

32-bit

```
function PEPrintControlsShowing (   
    printJob: Word;  
    var controlsShowing: Bool  
): Bool stdcall;
```

PEPrintReport

Prints the specified report to either the printer or to a preview window. This function establishes a Print-Only link where changes are made during runtime by other PE calls are ignored. Use any time you simply want to print a report from an application without giving the user the ability to customize the report.

```
short CRPE_API PEPrintReport (
    char FAR *reportFilePath,
    BOOL toDefaultPrinter,
    BOOL toWindow,
    char FAR *title,
    int left,
    int top,
    int width,
    int height,
    DWORD style,
    HWND parentWindow
);
```

Parameters

Parameter	Description
reportFilePath	Specifies a pointer to the null-terminated string that contains the name and path of the report you want to print.
toDefaultPrinter	Specifies whether or not the report is to be sent to the default printer.
toWindow	Specifies whether or not the report is to be displayed in the preview window.
title	Specifies a pointer to the null-terminated string that contains the title you want to appear on the title bar if you are printing the report to a window.
left	Specifies the x coordinate of the upper left hand corner of the preview window, in pixels.
top	Specifies the y coordinate of the upper left hand corner of the preview window, in pixels.
width	Specifies the width of the preview window, in pixels.
height	Specifies the height of the preview window, in pixels.
style	Specifies the style of the window being created. Style settings can be combined using the bitwise “OR” operator. Select your style from the list that appears with <i>PEOutputToWindow</i> , Page 257.

<i>Parameter</i>	<i>Description</i>
parentWindow	Specifies the handle to the Parent Window if the preview window is a child of that window.

Return Value

0 if the call is successful; returns an error code if an error occurs.

PEPrintReport (Other Declares)

VB Syntax

16-bit

```
Declare Function PEPrintReport Lib "crpe.dll" (ByVal RptName As String, ByVal Printer As Integer, ByVal Window As Integer, ByVal Title As String, ByVal Left As Integer, ByVal Top As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal Style As Long, ByVal ParentWindow As Integer) As Integer
```

32-bit

```
Declare Function PEPrintReport Lib "crpe32.dll" (ByVal RptName As String, ByVal Printer As Integer, ByVal Window As Integer, ByVal Title As String, ByVal Left As Integer, ByVal Top As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal style As Long, ByVal ParentWindow As Long) As Integer
```

Delphi Syntax

16-bit

```
function PEPrintReport (
  reportFilePath: PChar;
  toDefaultPrinter:
  toWindow: Bool;
  title: PChar;
  left: integer
  top: integer
  width: integer
  height: integer;
  style: longint;
  parentWindow: HWnd
  ): integer;
```

32-bit

```
function PEPrintReport (
    reportFilePath: PChar;
    toDefaultPrinter: Bool;
    toWindow: Bool;
    title: PChar;
    left: integer;
    top: integer;
    width: integer;
    height: integer;
    style: longint;
    parentWindow: HWnd
): Smallint stdcall;
```

dBASE for Windows Syntax

```
EXTERN CWORD PEPrintReport (CSTRING, CLOGICAL, CLOGICAL, CSTRING,
CWORD, CWORD, CWORD, CLONG, CHANDLE) CRPE.DLL
```

PEPrintWindow

Prints the report displayed in the preview window. This function can be used in a Custom-Print link to enable the user to preview the report in the preview window, and then, if everything looks satisfactory, to print the report to the printer (in response to a user event).

```
BOOL CRPE_API PEPrintWindow (
    short printJob,
    BOOL waitUntilDone
);
```

Parameter

Parameter	Description
printJob	Specifies the handle to the print job you want to print to a window.
waitUntilDone	Indicates whether or not the function is to return as soon as printing starts. Use one of the following values:

Value	Meaning
TRUE	The function returns when printing is finished. In version 1.1 and higher of CRPE, "waitUntilDone" must be True.

<i>Value</i>	<i>Meaning</i>
FALSE	The function returns as soon as printing starts.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEPrintWindow (Other Declares)

VB Syntax

16-bit

```
Declare Function PEPrintWindow Lib "crpe.dll" (ByVal printJob As Integer, ByVal WaitNoWait As Integer) As Integer
```

32-bit

```
Declare Function PEPrintWindow Lib "crpe32.dll" (ByVal printJob As Integer, ByVal WaitNoWait As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEPrintWindow (
  printJob: integer;
  waitUntilDone: Bool
): Bool;
```

32-bit

```
function PEPrintWindow (
  printJob: Word;
  waitUntilDone: Bool
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEPrintWindow (CWORD, CLOGICAL) CRPE.DLL
```

PESelectPrinter

Specifies a printer other than the default printer as a print destination. You can use this function to enable the user to select a printer other than the default printer at print time. One way of doing this is to have your application call the Print Setup common dialog box.

16-bit

```
BOOL CRPE_API PESelectPrinter (
    short printJob,
    char FAR *driverName,
    char FAR *printerName,
    char FAR *portName,
    DEVMODE FAR *mode
);
```

32-bit

```
BOOL CRPE_API PESelectPrinter (
    short printJob,
    char FAR *driverName,
    char FAR *printerName,
    char FAR *portName,
    DEVMODEA FAR *mode
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to select a printer.
driverName*	Specifies a pointer to a null-terminated string that contains the name of the printer driver for the selected printer.
printerName*	Specifies a pointer to a null-terminated string that contains the printer name for the selected printer.
portName*	Specifies a pointer to a null-terminated string that contains the port name for the port to which the selected printer is attached.
mode*	Specifies a pointer to the Windows API structure <i>DEVMODE</i> , Page 353 . If you are using UNICODE, this will be a <i>DEVMODEA</i> structure.

*The PRINTDLG structure, returned by the Windows API PrintDlg function, contains handles to DEVMODE and DEVNAMES structures. This information can be used to obtain driverName, printerName, portName, and mode values for PESelectPrinter.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- Your code must parse the return from the dialog box selection and insert the returned Printer Driver Name, Printer Name, and Port Name as parameters in the call.
- After selecting the printer with this call, you can direct the output to that printer (using *PEOutputToPrinter*, *Page 254*) or to the preview window (using *PEOutputToWindow*, *Page 257*).
- This call will override a printer selection that you built into the report at design time via the Seagate Crystal Reports Select Printer menu command.
- If you follow this call with the *PEOutputToWindow* call, the report appears in the preview window.
- To revert to the default printer, pass zeros (0) for each parameter.
- The driver name and printer name must exist on your machine.
- You can specify a different printer port than that assigned to the selected printer on your machine.
- For *mode*, use 0 for the default mode or create a *DEVMODE*, *Page 353*, structure to customize (if your development tool supports such a structure).
- This function should be called before *PESetPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.
- The *mode* parameter points to either a *DEVMODE* or *DEVMODEA* structure. If you are using UNICODE in your application, *DEVMODEA* will be used. Otherwise, *DEVMODE* is used.

PESelectPrinter (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for *crPESelectPrinter* in Developer's online Help.

Delphi Syntax

16-bit

```
function PESelectPrinter (
    printJob: integer;
    driverName,
```

```

    printerName,
    portName: PChar;
    mode: PDevMode
  ): Bool;

```

32-bit

```

function PESelectPrinter(
  printJob: Word;
  driverName: PChar;
  printerName: PChar;
  portName: PChar;
  mode: PDeviceModeA
): Bool stdcall;

```

PESetAreaFormat

Sets the area format settings for selected areas in the specified report to the values in *PESectionOptions*, *Page 444*. This function can be used to provide specialized formatting for printing invoices, form letters, printing to pre-printed forms, etc. It allows you to hide an area, insert a page break either before or after an area begins, reset the page number to 1 after a group value prints, prevent page breaks from spreading data from a single record over two pages, and to print group values only at the bottom of a page.

```

BOOL CRPE_API PESetAreaFormat (
  short printJob,
  short areaCode,
  PESectionOptions FAR *options
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job for which you want to set area formatting options.
areaCode	Specifies the code for the report area for which you want to set formatting options. See the information on area codes in <i>Working with section codes</i> , <i>Page 116</i> .
options	Specifies a pointer to <i>PESectionOptions</i> , <i>Page 444</i> . Use this structure to set your section options.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PESetAreaFormat*, *Page 348*, or the results may be inconsistent or unexpected.

PESetAreaFormat (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetAreaFormat Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal areaCode As Integer, options As  
PESectionOptions) As Integer
```

32-bit

```
Declare Function PESetAreaFormat Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, options As  
PESectionOptions) As Integer
```

Delphi Syntax

16-bit

```
function PESetAreaFormat (  
    printJob,  
    areaCode: Integer;  
    var options: PESectionOptions  
): Bool;
```

32-bit

```
function PESetAreaFormat (  
    printJob: Word;  
    areaCode: Integer;  
    options: PESectionOptions  
): Bool stdcall;
```

PESetAreaFormatFormula

Changes the specified area format formula to the formula string you supply as a parameter. This function will only change the text of a formula which already exists in the report; you can not use it to add a formula.

When you give the user the ability to change the formula at print time, your link must include code to replace formulaString with a user-generated value.

```
BOOL CRPE_API PESetAreaFormatFormula (
    short printJob,
    short areaCode,
    short formulaName,
    char FAR *formulaString
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set a new format formula string.
areaCode	Specifies the code for the report area for which you want to set formatting options. For information on area codes, see <i>Working with section codes, Page 116</i> .
formulaName	Specifies the name of the formatting formula for which you want to supply a new string. Use one of the following constants:

Constant	Value	Meaning
PE_FFN_SECTION _VISIBILITY	58	Specifies that the section be visible. If not visible, no drill down will be available.
PE_FFN_SHOW_AREA	59	Specifies that the area be visible. If not visible, drill down will be available.
PE_FFN_NEW_PAGE _BEFORE	60	Specifies that a new page will be printed before the section.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FFN_NEW_PAGE _AFTER	61	Specifies that a new page will be printed after the section.
PE_FFN_KEEP_TOGETHER	62	Specifies that the section will be kept together.
PE_FFN_SUPPRESS_BLANK _SECTION	63	Specifies that if the section is blank, it will not be printed.
PE_FFN_RESET_PAGE_N _AFTER	64	Specifies that page numbering will reset after the section.

<i>Parameter</i>	<i>Description</i>
formulaString	Specifies a pointer to the null-terminated string that you want to assign to the format formula.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- The formulaName constant PE_FFN_SECTION_VISIBILITY is supported for compatibility with earlier versions of Crystal Report Engine applications. This constant has the same value as PE_FFN_AREASECTION_VISIBILITY. All new Crystal Report Engine applications should use PE_FFN_AREASECTION_VISIBILITY.
- This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.
- Not all parameters apply to all areas.

PESetAreaFormatFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetAreaFormatFormula Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, ByVal  
formulaName As Integer, ByVal formulaString As String) As  
Integer
```

32-bit

```
Declare Function PESetAreaFormatFormula Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal areaCode As Integer, ByVal  
formulaName As Integer, ByVal formulaString As String) As  
Integer
```

Delphi Syntax

16-bit

```
function PESetAreaFormatFormula(  
    printJob: integer;  
    areaCode: integer;  
    formulaName: integer;  
    formulaString: Pchar  
): Bool;
```

32-bit

```
function PESetAreaFormatFormula(  
    printJob: Word;  
    areaCode: Word;  
    formulaName: Word;  
    formulaString: Pchar  
): Bool stdcall;
```

PESetDialogParentWindow



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Specifies the handle for the parent window of CRPE dialog boxes (i.e., Print Progress dialog box).

```
BOOL CRPE_API PESetDialogParentWindow (
    short printJob,
    HWND parentWindow
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to specify a parent window.
parentWindow	Specifies the handle of the parent window.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PESetDialogParentWindow (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetDialogParentWindow Lib "crpe.dll" (ByVal
printJob As Integer, ByVal parentWindow As Long) As Integer
```

32-bit

```
Declare Function PESetDialogParentWindow Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal parentWindow As Long) As Integer
```

Delphi Syntax

16-bit

```
function PESetDialogParentWindow (
    printJob: integer;
    parentWindow: HWnd
): Bool;
```

32-bit

```
function PESetDialogParentWindow (
    printJob: Word;
    parentWindow: HWnd
): Bool stdcall;
```



PESetEventCallback



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

PESetEventCallback sets the event callback function for the specified job. CRPE can find certain events when something happens inside CRPE. CRPE will call the callback function and notify what kind of event has or is about to occur. Within PESetEventCallback, the user can interpret the event ID and perform the proper process.

16-bit

```
BOOL CRPE_API PESetEventCallback (
    short printJob,
    BOOL (CALLBACK_export *callbackProc)
        (short eventID, void *param, void *userData)
    void *userData
);
```

32-bit

```
BOOL CRPE_API PESetEventCallback (
    short printJob,
    BOOL (CALLBACK *callbackProc)
        (short eventID, void *param, void *userData)
    void *userData
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the printJob for which you want to create an Event procedure.
callbackProc	The CALLBACK procedure that will handle your Crystal Report Engine events. This should be a pointer to a standard Windows CALLBACK procedure. Refer to the Windows SDK for information on creating CALLBACK procedures.
userData	Any information you want to pass to the Event handling CALLBACK procedure. The pointer will be available in the userData member of the procedure. This value can be 0.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- If PESetEventCallback returns TRUE (1), a CRPE default action will be provided. If PESetEventCallback returns FALSE (0), a CRPE default action will not be used. The user should be responsible for providing default behavior. For some events, the PESetEventCallback return value is ignored. The following gives the description of different events supported by CRPE:

PE_CLOSE_PRINT_WINDOW_EVENT

- **Called**

Before the preview window is closed.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, [Page 409](#).

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_ACTIVATE_PRINT_WINDOW_EVENT

- **Called**

Before the preview window becomes active.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, [Page 409](#).

- **Return Value**

Ignored

PE_DEACTIVATE_PRINT_WINDOW_EVENT

- **Called**

Before the preview window becomes in active.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

Ignored

PE_PRINT_BUTTON_CLICKED_EVENT

- **Called**

After the Print button is clicked; before printing process starts.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_EXPORT_BUTTON_CLICK_EVENT

- **Called**

After Export button is clicked; before export process starts.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_ZOOM_LEVEL_CHANGING_EVENT

- **Called**

After changing zoom control; before changing preview zoom level.

- **Parameter**

Pointer to *PEZoomLevelChangingEventInfo*, *Page 473*.

- **Return Value**

Ignored

PE_FIRST_PAGE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the first page button; before going to the first page.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_PREVIOUS_PAGE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the previous page button; before going to the previous page.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_NEXT_PAGE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the next page button, before going to the next page.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_LAST_PAGE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the last page button; before going to the last page.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_CANCEL_BUTTON_CLICKED_EVENT

- **Called**

After clicking the cancel button; before canceling the printing or reading the database.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_CLOSE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the close button; before closing the preview window.

- **Parameter**

Pointer to *PECloseButtonClickedEventInfo*, *Page 391*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

- **Remarks**

If FALSE (0) is returned before actually closing the preview window, a PE_CLOSE_PRINT_WINDOW_EVENT is fired.

PE_SEARCH_BUTTON_CLICKED_EVENT

- **Called**

After the search button is clicked; before the search starts.

- **Parameter**

Pointer to *PESearchButtonClickedEventInfo*, *Page 442*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_GROUP_TREE_BUTTON_CLICKED_EVENT

- **Called**

After clicking the group tree button; before showing or hiding the group tree.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

Ignored

PE_PRINT_SETUP_BUTTON_CLICKED_EVENT

- **Called**

After clicking Print Setup button; before showing the Print Setup dialog box.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_REFRESH_BUTTON_CLICKED_EVENT

- **Called**

After clicking the Refresh button; before refreshing the data.

- **Parameter**

Pointer to *PEGeneralPrintWindowEventInfo*, *Page 409*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_SHOW_GROUP_EVENT

- **Called**

After clicking one of the group tree nodes; before showing that group.

- **Parameter**

Pointer to *PEShowGroupEventInfo*, *Page 450*.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_DRILL_ON_GROUP_EVENT

- **Called**

After clicking on one of the group tree nodes, double-clicking or Ctrl-clicking a node with the magnify glass cursor, or double-clicking one of the groups in the preview window; before showing the group.

- **Parameter**

Pointer to *PEDrillOnGroupEventInfo*, Page 394.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_DRILL_ON_DETAIL_EVENT

- **Called**

After double-clicking one of the detail areas in the preview window.

- **Parameter**

Pointer to *PEDrillOnDetailEventInfo*, Page 392.

- **Return Value**

Ignored

PE_READING_RECORDS_EVENT

- **Called**

This event is fired during a reading database or regenerating saved data process. It is fired after a specified amount of time, not after reading every record.

- **Parameter**

Pointer to *PEReadingRecordsEventInfo*, Page 438.

- **Return Value**

Ignored

PE_START_EVENT

- **Called**

Before the Report Engine starts a process. A process can be printing to printer, exporting, printing to a window, or generating pages when navigating through the preview window.

- **Parameter**

Pointer to *PEStartEventInfo*, Page 451.

- **Return Value**

TRUE (1) to use the default action. FALSE (0) to cancel the default action.

PE_END_EVENT

- **Called**

Whenever a process has finished. Used in conjunction with PE_START_EVENT.

- **Parameter**

Pointer to *PESetEventInfo*, *Page 453*.

- **Return Value**

Ignored

- Each job can have only one callback function.
- The Event procedure functions passed to the *callbackProc* parameter should be a standard Windows CALLBACK procedure. Refer to documentation on the Windows API for information on creating CALLBACK procedures.
- If you need to pass data to the CALLBACK procedure using the *userData* parameter of PESetEventCallback, be sure the memory allocated for the data does not fall out of scope or is deallocated before the CALLBACK function is called by Windows. If this happens, the data will be unavailable, an errors may occur in your application.
- For a complete example of how to use this function, see the *Handling Preview Window Events, Page 71*.

PESetEventCallback (Other Declares)

VB Syntax

This function is not available in Visual Basic.

Delphi Syntax

16-bit

```
function PESetEventCallback(
    printJob: Integer;
    callbackProc: pointer
        {callback Function should be of form:
         Function callbacProc(eventID: smallint;
                               param2: pointer;
                               param3: pointer)}
    ): Bool; stdcall;
```

32-bit

```
function PESetEventCallback(
    printJob: Word;
    callbackProc: pointer
        {Callback Function should be of form:
        Function callbacProc(eventID: smallint;
            param: pointer;
            userData: pointer)}
): Bool stdcall;
```

PESetFont



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the font for field and/or text characters in the report section(s) specified. Use any time you need to change a default font at runtime in response to user input or to specify a built-in printer font.

```
BOOL CRPE_API PESetFont (
    short printJob,
    short sectionCode,
    short scopeCode,
    char FAR *faceName,
    short fontFamily,
    short fontPitch,
    short charSet,
    short pointSize,
    short isItalic,
    short isUnderlined,
    short isStruckOut,
    short weight
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to select a font.
sectionCode	Specifies the code for the report section(s) for which you want to select a font. See <i>Working with section codes, Page 116</i> .

<i>Parameter</i>	<i>Description</i>
scopeCode	Specifies whether the font selected is to apply to fields, to text, or to both. To specify both fields and text, use the OR operator. Select your code from the following:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FIELDS	0x0001	Sets the default font for all field values in the report section specified.
PE_TEXT	0x0002	Sets the default font for all text (that has not been entered as a field value) in the report section specified.

<i>Parameter</i>	<i>Description</i>
faceName	Specifies the actual face name of the font you want to use. The face name you pass can typically come from a Font dialog box, be hard coded in the application or be chosen by the application from the fonts supported on the printer. Example: "Times New Roman". Pass 0 for no change.
fontFamily	Specifies the font family for the font you want to use. Use one of the following constant values:

<i>Constant</i>	<i>Meaning</i>
FF_DONTCARE	No change.
FF_ROMAN	Variable pitch font with serifs.
FF_SWISS	Fixed pitch font without serifs.
FF_MODERN	Fixed-pitch font, with or without serifs.
FF_SCRIPT	Handwriting-like font.
FF_DECORATIVE	Fancy display font.

<i>Parameter</i>	<i>Description</i>
fontPitch	Specifies the font pitch you wish to use. Use a constant value for the font pitch as defined in WINDOWS.H. Use DEFAULT_PITCH if you wish to retain the current default.

<i>Constant</i>	<i>Value</i>
DEFAULT_PITCH	0X00
FIXED_PITCH	0X01
VARIABLE_PITCH	0X02

<i>Parameter</i>	<i>Description</i>
charSet	Specifies the character set you wish to use. Use a constant value for the character set as defined in WINDOWS.H. Use DEFAULT_CHARSET if you wish to retain the current default.

<i>Constant</i>	<i>Value</i>
ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
HANGEUL_CHARSET	129
CHINESEBIG5_CHARSET	136
OEM_CHARSET	255

<i>Parameter</i>	<i>Description</i>
pointSize	Specifies the desired point size for the selected font. Enter 0 for no change.
isItalic	Specifies whether the font selected should be italicized. Use 1 for True (the font is italic), 0 for False (the font is not italic), or PE_UNCHANGED (if there is no change from the current default).
isUnderlined	Specifies whether the font selected should be underlined. Use 1 for True (the font is underlined), 0 for False (the font is not underlined), or PE_UNCHANGED (if there is no change from the current default).
isStruckOut	Specifies whether the font selected should be struck out. Use 1 for True (the font is struck out), 0 for False (the font is not struck out), or PE_UNCHANGED (if there is no change from the default).
weight	Specifies the weight of the font. Use a constant value from the weight values defined in WINDOWS.H. Use 0 if you wish to retain the current default.

<i>Constant</i>	<i>Value</i>
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_HEAVY	900
FW_ULTRALIGHT	FW_EXTRALIGHT
FW_REGULAR	FW_NORMAL
FW_DEMIBOLD	FW_SEMIBOLD
FW_ULTRABOLD	FW_EXTRABOLD
FW_BLACK	FW_HEAVY

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This command includes a number of parameters:

- For the fontFamily, fontPitch, charSet, and weight parameters, use constant values from the font family, pitch, character set, and width defined in WINDOWS.H. Use 0 for each parameter that is not to be changed from the current default.
- For the faceName parameter, enter the actual name of the font. Enter 0 for no change.
- faceName, fontFamily, fontPitch, and charSet should all be specified whenever one of these parameters is specified. Use fontFamily = FF_DONTCARE (0), fontPitch = DEFAULT_PITCH (0), or charSet = DEFAULT_CHARSET (1) to leave the default values unchanged.
- This function should be called before *PEStartPrintJob*, Page 348, or the results may be inconsistent or unexpected.

Related Menu Commands

File | Options | Fonts Tab

PESetFont (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetFont Lib "crpe.dll" (ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal ScopeCode As Integer, ByVal FaceName As String, ByVal FontFamily As Integer, ByVal FontPitch As Integer, ByVal CharSet As Integer, ByVal PointSize As Integer, ByVal isItalic As Integer, ByVal isUnderlined As Integer, ByVal isStruckOut As Integer, ByVal Weight As Integer) As Integer
```

32-bit

```
Declare Function PESetFont Lib "crpe32.dll" (ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal ScopeCode As Integer, ByVal FaceName As String, ByVal FontFamily As Integer, ByVal FontPitch As Integer, ByVal CharSet As Integer, ByVal PointSize As Integer, ByVal isItalic As Integer, ByVal isUnderlined As Integer, ByVal isStruckOut As Integer, ByVal Weight As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetFont (
    printJob: integer;
    sectionCode: integer;
    scopeCode: integer;
    faceName: PChar;
    fontFamily: integer;
    fontPitch: integer;
    charSet: integer;
    pointSize: integer;
    isItalic: integer;
    isUnderlined: integer;
    isStruckOut: integer;
    weight: integer
): Boolean;
```

32-bit

```
function PESetFont(  
    printJob: Word;  
    sectionCode: integer;  
    scopeCode: integer;  
    faceName: PChar;  
    fontFamily: integer;  
    fontPitch: integer;  
    charSet: integer;  
    pointSize: integer;  
    isItalic: integer;  
    isUnderlined: integer;  
    isStruckOut: integer;  
    weight: integer  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetFont (CWORD, CWORD, CWORD, CSTRING, CWORD, CWORD,  
CWORD, CWORD, CWORD, CWORD, CWORD) CRPE.DLL
```

PESetFormula

Changes the specified formula to the formula string you supply as a parameter. This function will only change the text of a formula which already exists in the report; you can not use it to add a formula. This function can be used by itself to replace the formula string for a known formula.

This function can also be used as one of a series of functions (*PEGGetFormula*, *Page 156*, *PEGGetHandleString*, *Page 170*, or *PESetFormula*). The series can be used in a Custom-Print link to identify and then change an existing formula at print time in response to a user selection.

When you give the user the ability to change the formula at print time, your link must include code to replace formulaString with a user-generated value.

```
BOOL CRPE_API PESetFormula (  
    short printJob,  
    char *formulaName,  
    char FAR *formulaString  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set a new formula string.
formulaName	Specifies a pointer to the null-terminated string that contains the name of the formula for which you want to set a new formula string.
formulaString	Specifies a pointer to the null-terminated string that you want to replace the existing formula string.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the named formula does not exist or if there is an error in the new formula.

NOTE: If the formula does not exist, the program generates a PE_ERR_BADFORMULANAME code. If there is an error in the formula, the program generates a PE_ERR_BADFORMULATEXT code. Search for Report Engine Error Codes in Developer's online Help.

Remarks

- This function should be called before *PESetFormula*, *Page 348*, or the results may be inconsistent or unexpected.
- You can not use this function to set conditional formulas.

PESetFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetFormula Lib "crpe.dll" (ByVal printJob As Integer, ByVal FormulaName As String, ByVal FormulaString As String) As Integer
```

32-bit

```
Declare Function PESetFormula Lib "crpe32.dll" (ByVal printJob As Integer, ByVal FormulaName As String, ByVal FormulaString As String) As Integer
```

Delphi Syntax

16-bit

```
function PESetFormula (
    printJob: integer;
    formulaName: PChar;
    formulaString: PChar
): Bool;
```

32-bit

```
function PESetFormula (
    printJob: Word;
    formulaName: PChar;
    formulaString: PChar
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetFormula (CWORD, CSTRING, CSTRING) CRPE.DLL
```

PESetGraphData

Changes the data on which the specified graph in the specified section is based. Use this function to change what data is used from your report to create a specified graph. This information includes the groups used to create the rows and columns of the graph and the summary field used to set the values of the risers in the graph.

```
BOOL CRPE_API PESetGraphData (
    short printJob,
    short sectionCode,
    short graphN,
    struct PEGraphDataInfo FAR *graphDataInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for the report containing the graph that you are changing.
sectionCode	Specifies the code for the report section in which the graph appears. See <i>Working with section codes, Page 116</i> .

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you are changing. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphDataInfo	Specifies the pointer to <i>PEGraphDataInfo</i> , <i>Page 410</i> .

Return Value

TRUE (1) if the call is successful; FALSE(0) if an error occurs.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetGraphData (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGraphData Lib "crpe.dll" (ByVal printJob
As Integer, ByVal sectionCode As Integer, ByVal GraphN As
Integer, GraphDataInfo As PEGraphDataInfo) As Integer
```

32-bit

```
Declare Function PESetGraphData Lib "crpe32.dll" (ByVal printJob
As Integer, ByVal sectionCode As Integer, ByVal GraphN As
Integer, GraphDataInfo As PEGraphDataInfo) As Integer
```

Delphi Syntax

16-bit

```
function PESetGraphData (
  printJob: integer;
  sectionCode: integer;
  graphN: integer;
  var graphDataInfo: PEGraphDataInfo
): Boolean;
```

32-bit

```
function PESetGraphData (
    printJob: Word;
    sectionCode: integer;
    graphN: integer;
    graphDataInfo: PEGraphDataInfo
): Bool stdcall;
```

PESetGraphOptions

Changes the display options set for the specified graph in the specified section. Use this function to change any of several graph options. These options include the minimum and maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

```
BOOL CRPE_API PESetGraphOptions (
    short printJob,
    short sectionCode,
    short graphN,
    struct PEGraphOptions FAR *graphOptions
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for the report containing the graph you are changing.
sectionCode	Specifies the code for the report section in which the graph appears. See <i>Working with section codes, Page 116</i> .
graphN	Specifies which graph within the section you want to change. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphOptions	Specifies the pointer to <i>PEGraphOptions, Page 413</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PESetPrintJob, Page 348*, or the results may be inconsistent or unexpected.

PESetGraphOptions (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPESetGraphOptions in Developer's online Help.

Delphi Syntax

16-bit

```
function PESetGraphOptions (
    printJob: integer;
    sectionCode: integer;
    graphN: integer;
    var graphOptions: PEGraphOptions
): Bool;
```

32-bit

```
function PESetGraphOptions (
    printJob: Word;
    sectionCode: smallint;
    graphN: smallint;
    graphOptions: PEGraphOptions
): Bool stdcall;
```

PESetGraphText

Changes the identifying text that appears on the specified graph in the specified section. This function allows you to set or change the text that appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs only).

```
BOOL CRPE_API PESetGraphText (
    short printJob,
    short sectionCode,
    short graphN,
    struct PEGraphTextInfo FAR *graphTextInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for the report containing the graph that you are changing the text for.
sectionCode	Specifies the code for the report section in which the graph appears. See <i>Working with section codes, Page 116</i> .
graphN	Specifies which graph within the section you are changing the text for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.
graphTextInfo	Specifies the pointer to <i>PEGraphTextInfo, Page 415</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PESetPrintJob, Page 348*, or the results may be inconsistent or unexpected.

PESetGraphText (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGraphText Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphTextInfo As PEGraphTextInfo) As Integer
```

32-bit

```
Declare Function PESetGraphText Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphTextInfo As PEGraphTextInfo) As Integer
```

Delphi Syntax

16-bit

```
function PESetGraphText (
    printJob: integer;
    sectionCode: integer;
    graphN: integer;
    var graphTextInfo: PEGraphTextInfo
): Bool;
```

32-bit

```
function PESetGraphText (
    printJob: Word;
    sectionCode: smallint;
    graphN: smallint;
    graphTextInfo: PEGraphTextInfo
): Bool stdcall;
```

PESetGraphType

Changes the type of graph displayed for the specified graph in the specified section based on one of the available graph/chart types. Use this function to change the type of graph that is displayed in the report. There are many types of graphs and charts possible with Seagate Crystal Reports. This function allows you to change which type of graph or chart is being displayed.

```
BOOL CRPE_API PESetGraphType (
    short printJob,
    short sectionCode,
    short graphN,
    short FAR *graphType
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for the report containing the graph for which you want to change the type.
sectionCode	Specifies the code for the report section in which the graph appears. See <i>Working with section codes, Page 116</i> .

<i>Parameter</i>	<i>Description</i>		
graphN	Specifies which graph within the section for which you want to change the type. This value is 0 indexed. Within a section, graphs are numbered from top to bottom, first, then from left to right if they have the same top.		
graphType	Specifies a pointer to the type of graph you want to be displayed. Use one of the following values:		
<i>Constant</i>	<i>Value</i>	<i>Meaning</i>	
PE_SIDE_BY_SIDE_BAR_GRAPH	0	Side By Side bar graph	
PE_STACKED_BAR_GRAPH	2	Stacked bar graph	
PE_PERCENT_BAR_GRAPH	3	Percent bar graph	
PE_FAKE_3D_SIDE_BY_SIDE_BAR_GRAPH	4	3D Side By Side bar graph	
PE_FAKE_3D_STACKED_BAR_GRAPH	5	3D Stacked bar graph	
PE_FAKE_3D_PERCENT_BAR_GRAPH	6	3D Percent bar graph	
PE_PIE_GRAPH	40	Pie graph	
PE_MULTIPLE_PIE_GRAPH	42	Multiple Pie graph	
PE_PROPORIONAL_MULTI_PIE_GRAPH	43	Weighted Pie graph	
PE_LINE_GRAPH	80	Line graph	
PE_AREA_GRAPH	120	Area graph	
PE_THREED_BAR_GRAPH	160	3D bar graph	
PE_USER_DEFINED_GRAPH	500	User Defined graph type	
PE_UNKNOWN_TYPE_GRAPH	1000	Unknown graph type	

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetGraphType (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGraphType Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphType As Integer) As Integer
```

32-bit

```
Declare Function PESetGraphType Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal sectionCode As Integer, ByVal GraphN As  
Integer, GraphType As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetGraphType (  
    printJob: integer;  
    sectionCode: integer;  
    graphN: integer;  
    var graphType: integer  
): Bool;
```

32-bit

```
function PESetGraphType (  
    printJob: Word;  
    sectionCode: smallint;  
    graphN: smallint;  
    graphType: smallint  
): Bool stdcall;
```

PESetGroupCondition



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Changes the group condition for a group section. Use this function whenever you want to change the grouping at print time, for example, to print one report grouped in several different ways.

```

BOOL CRPE_API PESetGroupCondition (
    short printJob,
    short sectionCode,
    char FAR *conditionField,
    short condition,
    short sortDirection
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job for which you want to change the group condition for a group section.
sectionCode	Specifies the code for the report section for which you want to set the group condition. See <i>Working with section codes, Page 116</i> .
conditionField	Specifies the pointer to the name of the field that triggers a summary whenever its value changes. This parameter is a result of calling <i>PEGetHandleString, Page 170</i> with conditionFieldHandle and conditionFieldLength, returned by <i>PESetGroupCondition, Page 296</i> .
condition	Specifies the condition (“weekly”, “monthly”, “change to Yes”, “change to No”, etc.) that needs to be met for Date and Boolean fields.

For all field types except Date and Boolean, use PE_GC_ANYCHANGE as the condition parameter.

<i>Constant (Date Fields Only)</i>	<i>Value</i>	<i>Meaning</i>
PE_GC_DAILY	0	Triggers a grouping every time the date changes.
PE_GC_WEEKLY	1	Triggers a grouping every time the date changes from one week to the next. (A week runs from Sunday through Saturday.)

<i>Constant (Date Fields Only)</i>	<i>Value</i>	<i>Meaning</i>
PE_GC_BIWEEKLY	2	Triggers a grouping every time the date changes from one two-week period to the next. (Weeks run from Sunday through Saturday.)
PE_GC_SEMIMONTHLY	3	Triggers a grouping every time the date changes from one half-month period to the next.
PE_GC_MONTHLY	4	Triggers a grouping every time the date changes from one month to the next.
PE_GC_QUARTERLY	5	Triggers a grouping every time the date changes from one calendar quarter to the next.
PE_GC_SEMIANNUALLY	6	Triggers a grouping every time the date changes from one half-year period to the next.
PE_GC_ANNUALLY	7	Triggers a grouping every time the date changes from one year to the next.

<i>Constant (Boolean Fields Only)</i>	<i>Value</i>	<i>Meaning</i>
PE_GC_TOYES	1	Triggers a grouping every time the sort and group by field value changes from No to Yes.
PE_GC_TONO	2	Triggers a grouping every time the sort and group by field value changes from Yes to No.
PE_GC_EVERYYES	3	Triggers a grouping every time the sort and group by field value is Yes.
PE_GC_EVERYNO	4	Triggers a grouping every time the sort and group by field value is No.
PE_GC_NEXTISYES	5	Triggers a grouping every time the next value in the sort and group by field is Yes.
PE_GC_NEXTISNO	6	Triggers a grouping every time the next value in the sort and group by field is No.

<i>Parameter</i>	<i>Description</i>
sortDirection	Specifies one of the following sort directions:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_SF_DESCENDING	0	Sorts data in descending order (Z to A, 9 to 1).
PE_SF_ASCENDING	1	Sorts data in ascending order (A to Z, 1 to 9).
PE_SF_ORIGINAL	2	Sorts data in its original order. (Group condition only.)
PE_SF_SPECIFIED	3	Sorts data in a specified order. (Group condition only.)

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- No default values are allowed. You must specify a value for all parameters when using this function.
- If you have a formula that references a summary field and you change the condition on the summary field without fixing the formula, you will get an error.
- This function should be called before *PESetGroupCondition*, *Page 348*, or the results may be inconsistent or unexpected.

Related Menu Commands

Edit | Group Section

PESetGroupCondition (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGroupCondition Lib "crpe.dll" (ByVal
printJob As Integer, ByVal sectionCode As Integer, ByVal
```

```
ConditionField As String, ByVal Condition As Integer, ByVal  
SortDirection As Integer) As Integer
```

32-bit

```
Declare Function PESetGroupCondition Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, ByVal  
ConditionField As String, ByVal Condition As Integer, ByVal  
SortDirection As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetGroupCondition (  
    printJob: integer;  
    sectionCode: integer;  
    conditionField: PChar;  
    condition: integer;  
    sortDirection: integer  
): Bool;
```

32-bit

```
function PESetGroupCondition (  
    printJob: Word;  
    sectionCode: smallint;  
    conditionField: PChar;  
    condition: smallint;  
    sortDirection: smallint  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetGroupCondition (CWORD, CWORD, CSTRING, CWORD,  
CWORD) CRPE.DLL
```

PESetGroupOptions



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets grouping options for the specified group.

```
BOOL CRPE_API PESetGroupOptions (
    short printJob,
    short groupN,
    PEGroupOptions FAR *groupOptions
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you wish to set grouping options.
groupN	Specifies the group level number starting from 0.
groupOptions	Specifies a pointer to <i>PEGroupOptions</i> , Page 418 .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function should be called before *PEStartPrintJob*, [Page 348](#), or the results may be inconsistent or unexpected.
- If you are using PESetGroupOptions to set the top/bottom N sort field, all the group sort fields related to the group will be deleted and a new one specified by the group options will be added.

PESetGroupOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGroupOptions Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal groupN As Integer, groupOptions As  
PEGroupOptions) As Integer
```

32-bit

```
Declare Function PESetGroupOptions Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal groupN As Integer, groupOptions As  
PEGroupOptions) As Integer
```

Delphi Syntax

16-bit

```
function PESetGroupOptions (  
    printJob: Word;  
    groupN: integer;  
    var groupOptions: PEGroupOptions  
): Bool;
```

32-bit

```
function PESetGroupOptions (  
    printJob: Word;  
    groupN: smallint;  
    var groupOptions: PEGroupOptions  
): Bool; stdcall;
```

PESetGroupSelectionFormula

Changes the group selection formula to the formula string you supply as a parameter. This function can be used by itself to replace an existing group selection formula.

This function can also be used as one of a series of functions (*PEGetGroupSelectionFormula*, *Page 168*, *PEGetHandleString*, *Page 170*, or *PESetGroupSelectionFormula*). The series can be used in a custom-print link to identify and then change an existing group selection formula at print time in response to a user selection.

When you give the user the ability to change the group selection formula at print time, your link must include code to replace formulaString with a user-generated value.

```
BOOL CRPE_API PESetGroupSelectionFormula (
    short printJob,
    char FAR *formulaString
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to set a new group selection formula.
formulaString	Specifies a pointer to the null-terminated string you want to assign to the group selection formula.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails or if there is an error in the new formula.

NOTE: If there is an error in the formula, the program generates a PE_ERR_BADFORMULATEXT code. Search for Report Engine Error Codes in Developer's online Help.

NOTE: A group sort field is a summary value or a summarized field that determines the sort order of groups.

Remarks

This function should be called before *PESetGroupSelectionFormula*, *Page 348*, or the results may be inconsistent or unexpected.

PESetGroupSelectionFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetGroupSelectionFormula Lib "crpe.dll"
    (ByVal printJob As Integer, ByVal formulaString As String) As
    Integer
```

32-bit

```
Declare Function PESetGroupSelectionFormula Lib "crpe32.dll"
(ByVal printJob As Integer, ByVal formulaString As String) As
Integer
```

Delphi Syntax

16-bit

```
function PESetGroupSelectionFormula (
    printJob: integer;
    formulaString: PChar
): Bool;
```

32-bit

```
function PESetGroupSelectionFormula (
    printJob: Word;
    formulaString: PChar
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetGroupSelectionFormula (CWORD, CSTRING) CRPE.DLL
```

PESetMargins



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the page margins for the specified report to the values you supply as parameters. Use this function any time you want to set the printer margins at runtime in response to user specifications.

```
BOOL CRPE_API PESetMargins (
    short printJob,
    short left,
    short right,
    short top,
    short bottom
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set new margins.
left	Specifies the left margin.*
right	Specifies the right margin.*
top	Specifies the top margin.*
bottom	Specifies the bottom margin.*

Value	Meaning
PM_SM_DEFAULT	You can use PM_SM_DEFAULT for any of the margin parameters. For each margin you specify in this way, the program will use the appropriate default margin for the currently selected printer.

*Left, right, top, and bottom values are all in twips. A twip is 1/1440 of an inch; there are 20 twips in a point. To set .5" margins, for example, you would enter the value 720.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetMargins (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetMargins Lib "crpe.dll" (ByVal printJob As Integer, ByVal leftMargin As Integer, ByVal rightMargin As Integer, ByVal topMargin As Integer, ByVal bottomMargin As Integer) As Integer
```

32-bit

```
Declare Function PESetMargins Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal leftMargin As Integer, ByVal rightMargin As  
Integer, ByVal topMargin As Integer, ByVal bottomMargin As  
Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetMargins (  
    printJob: integer;  
    left: Word;  
    right: Word;  
    top: Word;  
    bottom: Word  
): Bool;
```

32-bit

```
function PESetMargins (  
    printJob: Word;  
    left: Word;  
    right: Word;  
    top: Word;  
    bottom: Word  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetMargins (CWORD, CWORD, CWORD, CWORD,  
CRPE.DLL
```

PESetMinimumSectionHeight



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the section height to be the specified section height.

```
BOOL CRPE_API PESetMinimumSectionHeight (  
    short printJob,  
    short sectionCode,  
    short minimumHeight );
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to set the minimum section height.
sectionCode	Specifies the code for the report section(s) for which you want to set minimum section height. See <i>Working with section codes, Page 116</i> .
minimumHeight	Specifies the minimum section height (in twips) for the section(s) selected. A twip is 1/1440 inch - there are 20 twips to a point.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function should be called before *PEStartPrintJob, Page 348*, or the results may be inconsistent or unexpected.
- If the specified section height is less than the section's minimum section height, the section height will not be changed.

PESetMinimumSectionHeight (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetMinimumSectionHeight Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, ByVal  
minimumHeight As Integer) As Integer
```

32-bit

```
Declare Function PESetMinimumSectionHeight Lib "crpe32.dll"  
(ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal  
minimumHeight As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetMinimumSectionHeight (
    printJob,
    sectionCode,
    minimumHeight: integer
): Bool;
```

32-bit

```
function PESetMinimumSectionHeight (
    printJob: Word;
    sectionCode: smallint;
    minimumHeight: smallint
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetMinimumSectionHeight (CWORD, CWORD, CWORD)
CRPE.DLL
```

PESetNDetailCopies



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Prints multiple copies of the Details section of the report. You can use this function to print multiple copies of labels for a customer, multiple copies of a purchase order, or multiple copies of anything set up in the Details section of your report.

```
BOOL CRPE_API PESetNDetailCopies (
    short printJob,
    short nDetailCopies
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want set the number of copies to print.
nDetailCopies	Specifies the number of report copies you want to print.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.
- To change top/bottom N group sorting order, use *PESetGroupOptions*, *Page 302*.

PESetNDetailCopies (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetNDetailCopies Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal nDetailCopies As Integer) As Integer
```

32-bit

```
Declare Function PESetNDetailCopies Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal nDetailCopies As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetNDetailCopies (  
    printJob,  
    nCopies: integer  
) : Bool;
```

32-bit

```
function PESetNDetailCopies(  
    printJob: Word;  
    nDetailCopies: smallint;  
) : Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetNDetailCopies (CWORD, CWORD) CRPE.DLL
```

PESetNthGroupSortField

Sets one of the group sort fields in the specified report. This function can be used by itself to set a sort field and direction when there is not one already set, or to modify an existing sort field and direction when the sort field number, name, and direction are known.

The function can also be used as one of a series of functions: *PEGetNthGroupSortFields*, *Page 180* (called once), *PEGetNthGroupSortField*, *Page 193*, or *PEGetHandleString*, *Page 170*, (called as many times as needed to identify the correct group sort field), and *PESetNthGroupSortField* (called once, when the correct group sort field is identified). The series can be used in a Custom-Print link to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

When you give the user the ability to specify group sort field(s) and/or direction at print time, your link must include code to replace name, and/or direction with user-generated values.

```
BOOL CRPE_API PESetNthGroupSortField (
    short printJob,
    short sortFieldN,
    char FAR *name,
    short direction
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to set a group sort field.
sortFieldN	Specifies the number of the sort field you want to set. The first sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1. If the report has N sort fields, you can call the function with sortFieldN = N to add a new sort field to the end of the list of existing sort fields. If N=0, the function will create the first sort field.
name	Specifies a pointer to the null-terminated string that contains the name of the group sort field.
direction	Specifies one of the following sort directions:

Constant	Value	Meaning
PE_SF_DESCENDING	0	Sorts in Descending order (Z to A, 9 to 1).
PE_SF_ASCENDING	1	Sorts in Ascending order (A to Z, 1 to 9).

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_SF_ORIGINAL	2	Sort in original order. (Group condition only.)
PE_SF_SPECIFIED	3	Sorts in a specified order. (Group condition only.)

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetNthGroupSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetNthGroupSortField Lib "crpe.dll" (ByVal
printJob As Integer, ByVal sortFieldN As Integer, ByVal name As
String, ByVal direction As Integer) As Integer
```

32-bit

```
Declare Function PESetNthGroupSortField Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal sortFieldN As Integer, ByVal name As
String, ByVal direction As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetNthGroupSortField (
  printJob,
  sortFieldN: integer;
  name: PChar;
  direction: integer
): Boolean;
```

32-bit

```
function PESetNthGroupSortField (
    printJob: Word;
    sortFieldN: smallint;
    name: PChar;
    direction: smallint
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetNthGroupSortField (CWORD, CWORD, CSTRING, CWORD)
CRPE.DLL
```

PESetNthParam



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the value of a parameter in a stored procedure. Use this function when working with stored procedures in SQL database tables to set the value of a parameter in a stored procedure.

```
BOOL CRPE_API PESetNthParam (
    short printJob,
    short paramN,
    LPCSTR szParamValue
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job that contains the stored procedure which you want to set the value of a parameter.
paramN	0 indexed parameter number, indicates which parameter you want to set the value for.
szParamValue	String representation of the value you are assigning to the parameter.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- All parameter values must be passed as string values. If you wish to pass a numeric value, pass the value in quotes like this: “100”. The Crystal Report Engine will treat this as the value 100.
- Use *PEGetNParams*, *Page 184*, to obtain the number of parameters in a stored procedure.
- Use *PEGetNthParam*, *Page 195*, to obtain a particular parameter required by a stored procedure.
- If you want to set the parameter value for the stored procedure to NULL, pass the string constant “CRWNull” to the szParamValue argument.

Related Menu Commands

Database | Stored Procedure Parameters

PESetNthParam (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetNthParam Lib "crpe.dll" (ByVal printJob As Integer, ByVal paramN As Integer, ByVal ParamValue As String) As Integer
```

32-bit

```
Declare Function PESetNthParam Lib "crpe32.dll" (ByVal printJob As Integer, ByVal paramN As Integer, ByVal ParamValue As String) As Integer
```

Delphi Syntax

16-bit

```
function PESetNthParam(  
  printJob: integer;  
  paramN: integer;  
  ParamValue: PChar  
): Boolean;
```

32-bit

```
function PESetNthParam(
    printJob: Word;
    paramN: smallint;
    ParamValue: PChar
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetNthPARAM (CWORD, CWORD, CSTRING) CRPE.DLL
```

PESetNthParameterField



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets a value for the specified parameter field.

```
BOOL CRPE_API PESetNthParameterField (
    short printJob,
    short parameterN,
    struct PEParameterFieldInfo FAR *parameterInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set a parameter field value.
parameterN	Specifies the number of the parameter field in the report.
parameterInfo	Specifies a pointer to <i>PEParameterFieldInfo</i> , Page 427, used to pass the parameter field value information.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, Page 348, or the results may be inconsistent or unexpected.

PESetNthParameterField (Other Declares)

Visual Basic Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPESetNthParameterField in Developer's online Help.

Delphi Syntax

16-bit

```
function PESetNthParameterField (
    printJob: integer;
    varN: integer;
    var varInfo: PEParameterFieldInfo
): Bool;
```

32-bit

```
function PESetNthParameterField (
    printJob: Word;
    varN: Smallint;
    var varInfo: PEParameterFieldInfo
): Bool stdcall;
```

PESetNthSortField

Sets one of the sort fields in the specified report. This function can be used by itself to set a sort field/direction when there is not one already set, or to change a sort field/direction when the number and name of the sort field are known.

The function can also be used as one of a series of functions: *PEGetNSortFields*, *Page 187* (called once), *PEGetNthSortField*, *Page 200*, or *PEGetHandleString*, *Page 170*, (called together as many times as needed to identify the correct sort field), and *PESetNthSortField* (called once when the correct sort field is identified). The series can be used in a Custom-Print link to identify and then change an existing sort field and/or sort order at print time in response to a user selection.

When you give the user the ability to specify sort field(s) and/or direction at print time, your link must include code to replace name, and/or direction with user-generated values.

```
BOOL CRPE_API PESetNthSortField (
    short printJob,
    short sortFieldN,
```

```

    char FAR *name,
    short direction
);

```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job for which you want to set sort field information.
sortFieldN	Specifies the number of the sort fields you want to set. The first sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1 to replace an existing sort field. If the report has N sort fields, you can call the function with "sortFieldN" = N to add a new sort field to the end of the list of existing sort fields. If N=0, the function will add the first sort field.
name	Specifies a pointer to the null-terminated string that contains the name of the sort field.
direction	Specifies the sort direction. Use one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_SF_DESCENDING	0	Sorts in Descending order (Z to A, 9 to 1).
PE_SF_ASCENDING	1	Sorts in Ascending order (A to Z, 1 to 9).
PE_SF_ORIGINAL	2	Sort in original order. (Group condition only.)
PE_SF_SPECIFIED	3	Sorts in a specified order. (Group condition only.)

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, Page 348, or the results may be inconsistent or unexpected.

Related Menu Commands

Report | Record Sort Order

PESetNthSortField (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetNthSortField Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sortFieldN As Integer, ByVal name As  
String, ByVal direction As Integer) As Integer
```

32-bit

```
Declare Function PESetNthSortField Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sortFieldN As Integer, ByVal name As  
String, ByVal direction As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetNthSortField (  
    printJob: integer;  
    sortFieldN: integer;  
    name: PChar;  
    direction: integer  
): Bool;
```

32-bit

```
function PESetNthSortField (  
    printJob: Word;  
    sortFieldN: smallint;  
    name: PChar;  
    direction: smallint  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetNthSortField (CWORD, CWORD, CSTRING, CWORD)  
CRPE.DLL
```

PESetNthTableLocation

Sets the location for a selected table in the specified print job. This function is typically combined with *PEGetNthTableLocation*, *Page 203*, to identify the location of a table and then to change it.

```
BOOL CRPE_API PESetNthTableLocation (
    short printJob,
    short tableN,
    struct PETableLocation FAR *location
);
```

Parameters

<i>Parameter</i>	<i>Description</i>
printJob	Specifies the handle of the print job for which you want to set a table's location.
tableN	Specifies the number of the table for which you want to set a new location.
location	Specifies the pointer to <i>PETableLocation</i> , <i>Page 457</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

Database | Set Location

PESetNthTableLocation (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPESetNthTableLocation in Developer's online Help.

Delphi Syntax

16-bit

```
function PESetNthTableLocation (
    printJob,
    tableN: integer;
    var location: PETableLocation
) : Bool;
```

32-bit

```
function PESetNthTableLocation(
    printJob: Word;
    tableN: smallint;
    var location: PETableLocation
) : Bool stdcall;
```

PESetNthTableLogOnInfo

Sets the log on information for the specified print job to the values in *PELogOnInfo*, [Page 425](#).

```
BOOL CRPE_API PESetNthTableLogOnInfo (
    short printJob,
    short tableN,
    struct PELogOnInfo FAR *logOnInfo,
    BOOL propagateAcrossTables
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set table log on information.
tableN	Specifies the number of the table for which you want to set log on information. The first table is table 0. The last table is N-1.
logOnInfo	Specifies the pointer to the <i>PELogOnInfo</i> , Page 425 .
propagateAcrossTables	Indicates whether or not the program should apply the new log on information to any other tables in the report that had the same original server and database names as the specified table. You may use either of the following values:

<i>Value</i>	<i>Meaning</i>
TRUE	Program will apply log on information to all other tables that have the same original server and database names.
FALSE	Program will update only the selected table.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- See *PELogOnServer*, *Page 241*, comments for the logOnInfo struct definition.
- The program logs on when printing the report, but you must first set the correct log on information using PESetNthTableLogOnInfo.
- You must supply at least the password with this function with empty strings for the other parameters. Additionally, you can change the server, database, and/or user name by entering the appropriate values.
- Logging off is performed automatically when the print job is closed.
- PESetNthTableLogOnInfo allows you to override any of the log on parameters.
- When you create a report off a single database (for example, one .MDB file with multiple tables), set the propagateAcrossTables parameter to TRUE. This insures that the changes are made to all tables in the .MDB file (thus avoiding the necessity to code the changes for each table individually).
- This function can be used to set the location of an Essbase application and database used by a report. For complete information, see *PELogOnInfo*, *Page 425*.

PESetNthTableLogOnInfo (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPESetNthTableLogOnInfo in Developer's online Help.

Delphi Syntax

16-bit

```
function PESetNthTableLogOnInfo (
    printJob,
    tableN: integer;
    var logOnInfo: PELogOnInfo;
    propagateAcrossTables: Bool
  ): Bool;
```

32-bit

```
function PESetNthTableLogOnInfo (
    printJob: Word;
    tableN: smallint;
    var logOnInfo: PELogOnInfo;
    propagateAcrossTables: Bool
  ): Bool stdcall;
```

PESetNthTableSessionInfo

Sets the specified session information when opening a Microsoft Access table. Many Microsoft Access database tables require that a session be opened before the table can be used. Use PESetNthTableSessionInfo to open the session.

```
BOOL CRPE_API PESetNthTableSessionInfo (
    short printJob,
    short tableN,
    struct PESessionInfo FAR *sessionInfo,
    BOOL propagateAcrossTables
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to change the MS Access session information.
tableN	0 based table number specifying which table the session is being opened for.
sessionInfo	Refer to <i>PESessionInfo</i> , Page 448, for further information.
propagateAcrossTables	TRUE or FALSE value indicating whether the session information should be used for opening all tables being used in the report.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- This function is only applicable for secured MS Access databases which require a session to be opened before the database is accessed.
- In Microsoft Access 95 and later, an Access database can have session security (also known as user-level security), database-level security, or both. If the Access database contains only session security, simply pass the session password to the Password member of the PESessionInfo structure passed to the sessionInfo parameter. If the Access database contains database-level security, use a newline character, '\n' (ASCII character 10) followed by the database-level password. For example:

```
"\ndbpassword"
```

If the Access database contains both session security and database-level security, use the session password followed by the newline character and the database password:

```
"sesspswd\ndbpassword"
```

Alternately, database-level security can also be handled by assigning the database-level password to the Password member of the *PELogOnInfo*, [Page 425](#), structure and calling the *PESetNthTableLogOnInfo*, [Page 320](#), function.

PESetNthTableSessionInfo (Other Declares)

VB Syntax

This function can not be called directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crPESetNthTableSessionInfo in Developer's online Help.

Delphi Syntax

16-bit

```
function PESetNthTableSessionInfo (
  printJob: integer;
  tableN: integer;
  var sessionInfo: PESessionInfo;
  propagateAcrossTables: Boolean
): Boolean;
```

32-bit

```
function PESetNthTableSessionInfo (
    printJob: Word;
    tableN: smallint;
    var sessionInfo: PESessionInfo;
    propagateAcrossTables: Bool
): Bool stdcall;
```

PESetPrintDate

Sets a print date that may be different than the system calendar date. Use this function any time you want to show a print date (or use a print date in formulas) other than the actual date of printing.

```
BOOL CRPE_API PESetPrintDate (
    short printJob,
    short year,
    short month,
    short day
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set the print date.
year	Specifies the year component of the print date. Enter a 4 digit year value (1994, 1993, etc.).
month	Specifies the month component of the print date. Months are numbered from 1 to 12, where January = 1 and December = 12. To use July as the print month, for example, you would enter the value 7.
day	Specifies the day component of the print date. Enter the actual day of the month you want to use (7, 18, 28, etc.).

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- You change the print date, typically, when you want to run the report today yet have it appear to have been run on a different date. An example would be, if you were out of town on the last day of the previous month and you later want to run a report for that month and make it appear as if it were run on the last day of the month.
- This function should be called before *PESetPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

Related Menu Commands

Report | Set Print Date

PESetPrintDate (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetPrintDate Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal Date_Year As Integer, ByVal Date_Month As  
Integer, ByVal Date_Day As Integer) As Integer
```

32-bit

```
Declare Function PESetPrintDate Lib "crpe32.dll" (ByVal printJob  
As Integer, ByVal Date_Year As Integer, ByVal Date_Month As  
Integer, ByVal Date_Day As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESetPrintDate (   
    printJob: integer;  
    year: integer;  
    month: integer;  
    day: integer  
): Bool;
```

32-bit

```
function PESetPrintDate (
    printJob: Word;
    year: smallint;
    month: smallint;
    day: smallint
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetPrintDate (CWORD, CWORD, CWORD, CWORD) CRPE.DLL
```

PESetPrintOptions



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the print options for the report to the values supplied in *PEPrintOptions*, Page 435. Use this function any time you want to set the starting page number, the ending page number, the number of report copies, and/or collation instructions for a print job at runtime in response to user specifications.

```
BOOL CRPE_API PESetPrintOptions (
    short printJob,
    struct PEPrintOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set printing options.
options	Specifies the pointer to <i>PEPrintOptions</i> , Page 435. If this parameter is set to 0 (null), the function prompts the user for these options. Using this, you can get the behavior of the print-to-printer button in the preview window by calling PESetPrintOptions with a null pointer and then calling <i>PEPrintWindow</i> , Page 265.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Related Menu Commands

File | Print | Printer

PESetPrintOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetPrintOptions Lib "crpe.dll" (ByVal  
printJob As Integer, Options As PEPrintOptions) As Integer
```

32-bit

```
Declare Function PESetPrintOptions Lib "crpe32.dll" (ByVal  
printJob As Integer, Options As PEPrintOptions) As Integer
```

Delphi Syntax

16-bit

```
function PESetPrintOptions (   
    printJob: integer;  
    var options: PEPrintOptions  
): Bool;
```

32-bit

```
function PESetPrintOptions (   
    printJob: Word;  
    var options: PEPrintOptions  
): Bool stdcall;
```

PESetReportSummaryInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

The PESetReportSummaryInfo function is used to set report summary information. Report summary information corresponds to the Summary Info dialog box in Seagate Crystal Reports.

```

BOOL CRPE_API PESetReportSummaryInfo (
    short printJob,
    PEReportSummaryInfo FAR *summaryInfo
) ;

```

Parameters

Parameter	Description
printJob	Specifies the print job for which you wish to set report summary information.
summaryInfo	Specifies the pointer to <i>PEReportSummaryInfo</i> , <i>Page 440</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetReportSummaryInfo (Other Declares)

VB Syntax

16-bit

```

Declare Function PESetReportSummaryInfo Lib "crpe.dll" (ByVal
printJob as Integer, summaryInfo as PEReportSummaryInfo) as
Integer

```

32-bit

```

Declare Function PESetReportSummaryInfo Lib "crpe32.dll" (ByVal
printJob as Integer, summaryInfo as PEReportSummaryInfo) as
Integer

```

Delphi Syntax

16-bit

```
function PESetReportSummaryInfo (
    printJob: Word;
    var summaryInfo: PEReportSummaryInfo
): Bool;
```

32-bit

```
function PESetReportSummaryInfo (
    printJob: Word;
    var summaryInfo: PEReportSummaryInfo
): Bool; stdcall
```

PESetReportTitle



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Changes the report title in the report summary information. This function can also be used as one of a series of functions (*PEGetReportTitle*, *Page 212*, *PEGetHandleString*, *Page 170*, or *REFSetReportTitle*). This series can be used in a Custom-Print link to identify and then change an existing report title at print time in response to a user selection.

```
BOOL CRPE_API PESetReportTitle (
    short printJob,
    char FAR *title
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set the report title.
title	Specifies a null-terminated string containing the new title you want to assign to the report.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- When you give the user the ability to change the report title at print time, your link must include code to replace texturing with a user-generated value.
- This function should be called before *PESetReportTitle*, *Page 348*, or the results may be inconsistent or unexpected.
- You can also use *PESetSummaryInfo* to change the report title and other summary information.

Related Menu Commands

Report | Report Title

PESetReportTitle (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetReportTitle Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal Title As String) As Integer
```

32-bit

```
Declare Function PESetReportTitle Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal Title As String) As Integer
```

Delphi Syntax

16-bit

```
function PESetReportTitle (  
    printJob: integer;  
    title: PChar  
): Bool;
```

32-bit

```
function PESetReportTitle (  
    printJob: Word;  
    title: PChar  
): Bool stdcall;
```

dBASE for Windows Syntax

EXTERN CLOGICAL **PESetReportTitle** (CWORD, CSTRING) CRPE.DLL

PESetSectionFormat



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets the section format settings for selected sections in the specified report to the values in *PESectionOptions*, *Page 444*. This function can be used to provide specialized formatting for printing invoices, form letters, printing to pre-printed forms, etc. It allows you to hide a section, insert a page break either before or after a section begins, reset the page number to 1 after a group value prints, prevent page breaks from spreading data from a single record over two pages, and to print group values only at the bottom of a page.

```
BOOL CRPE_API PESetSectionFormat (
    short printJob,
    short sectionCode,
    PESectionOptions FAR *options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set section formatting options.
sectionCode	Specifies the code for the report section(s) for which you want to set formatting options. See <i>Working with section codes</i> , <i>Page 116</i> .
options	Specifies the pointer to <i>PESectionOptions</i> , <i>Page 444</i> . Use this structure to set your section options.

There can be multiple sections in an area. For more information on Section Codes, see *Working with section codes*, *Page 116*.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function should be called before *PESetSectionFormat*, *Page 348*, or the results may be inconsistent or unexpected.

Related Menu Commands

Format | Section

PESetSectionFormat (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetSectionFormat Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, Options As  
PESectionOptions) As Integer
```

32-bit

```
Declare Function PESetSectionFormat Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, Options As  
PESectionOptions) As Integer
```

Delphi Syntax

16-bit

```
function PESetSectionFormat (   
    printJob: integer;  
    sectionCode: integer;  
    options: PESectionOptions  
  ): Bool;
```

32-bit

```
function PESetSectionFormat (   
    printJob: Word;  
    sectionCode: smallint;  
    var options: PESectionOptions  
  ): Bool stdcall;
```

PESetSectionFormatFormula

Changes the specified section format formula to the formula string you supply as a parameter. This function will only change the text of a formula which already exists in the report; you can not use it to add a formula.

When you give the user the ability to change the formula at print time, your link must include code to replace formulaString with a user-generated value.

```
BOOL CRPE_API PESetSectionFormatFormula (
    short printJob,
    short sectionCode,
    short formulaName,
    char FAR *formulaString
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set a new selection formula string.
sectionCode	Specifies the code for the report section(s) for which you want to set formatting options. See <i>Working with section codes, Page 116</i> .
formulaName	Specifies the name of the formula for which you want to supply a new string. Use one of the following constants:

Constant	Value	Meaning
PE_FFN_SECTION _VISIBILITY	58	Specifies that the section be visible. If not visible, no drill down will be available.
PE_FFN_SHOW_AREA	59	Specifies that the area be visible. If not visible, drill down will be available.
PE_FFN_NEW_PAGE _BEFORE	60	Specifies that a new page will be printed before the section.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_FFN_NEW_PAGE _AFTER	61	Specifies that a new page will be printed after the section.
PE_FFN_KEEP_TOGETHER	62	Specifies that the section will be kept together.
PE_FFN_SUPPRESS_BLANK _SECTION	63	Specifies that if the section is blank, it will not be printed.
PE_FFN_RESET_PAGE_N _AFTER	64	Specifies that page numbering will reset after the section.
PE_FFN_PRINT_AT _BOTTOM_OF_PAGE	65	Specifies that the section will be printed at the bottom of the page.
PE_FFN_UNDERLAY _SECTION	66	Specifies that the section will overlay any sections following.
PE_FFN_SECTION_BACK _COLOUR	67	Specifies the background color of the section.

<i>Parameter</i>	<i>Description</i>
formulaString	Specifies a pointer to the null-terminated string that you want to assign to the format formula.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails or if there is an error in the new formula.

NOTE: If the formula name passed in does not exist, or it does not apply to the section, the program generates a PE_ERR_BADFORMULANAME code. If there is an error in the formula, the program generates a PE_ERR_BADFORMULATEXT code. Search for Report Engine Error Codes in Developer's online Help.

Remarks

- This function should be called before *PESetSectionFormatFormula*, *Page 348*, or the results may be inconsistent or unexpected.
- Not all formula names can be applied to all sections.

PESetSectionFormatFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetSectionFormatFormula Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal sectionCode As Integer, ByVal  
formulaName As Integer, ByVal FormulaString As String) As  
Integer
```

32-bit

```
Declare Function PESetSectionFormatFormula Lib "crpe32.dll"  
(ByVal printJob As Integer, ByVal sectionCode As Integer, ByVal  
formulaName As Integer, ByVal FormulaString As String) As  
Integer
```

Delphi Syntax

16-bit

```
function PESetSectionFormatFormula (  
  printJob: integer;  
  sectionCode: integer;  
  formulaName: integer;  
  formulaString: PChar  
): Bool;
```

32-bit

```
function PESetSectionFormatFormula (   
  printJob: Word;  
  sectionCode: smallint;  
  formulaName: smallint;  
  formulaString: PChar  
): Bool stdcall;
```

PESetSelectionFormula

Changes the selection formula to the formula string you supply as a parameter. This function can be used by itself to replace a known record selection formula.

The function can also be used as one of a series of functions (*PEGetSelectionFormula*, *Page 224*, *PEGetHandleString*, *Page 170*, or *PESetSelectionFormula*). The series can be used in a Custom-Print link to identify and then change an existing record selection formula at print time in response to a user selection.

When you give the user the ability to change the record selection formula at print time, your link must include code to replace *formulaString* with a user-generated value.

```
BOOL CRPE_API PESetSelectionFormula (
    short printJob,
    const char FAR *formulaString
);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to set a new selection formula string.
formulaString	Specifies a pointer to the null-terminated string that you want to assign to the selection formula.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails or if there is an error in the new formula.

NOTE: If there is an error in the formula, the program generates a PE_ERR_BADFORMULATEXT code. Search for Report Engine Error Codes in Developer's online Help.

Remarks

This function should be called before *PEStartPrintJob*, *Page 348*, or the results may be inconsistent or unexpected.

PESetSelectionFormula (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetSelectionFormula Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal FormulaString As String) As Integer
```

32-bit

```
Declare Function PESetSelectionFormula Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal FormulaString As String) As Integer
```

Delphi Syntax

16-bit

```
function PESetSelectionFormula (  
    printJob: integer;  
    formulaString: PChar  
): Bool;
```

32-bit

```
function PESetSelectionFormula (  
    printJob: Word;  
    formulaString: PChar  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetSelectionFormula (CWORD, CSTRING) CRPE.DLL
```

PESetSQLQuery



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Changes the SQL query to the query string you supply as a parameter. Use this function to update the SQL query that will be used to print the report, typically to add optimizations to the WHERE clause.

```

BOOL CRPE_API PESetSQLQuery (
    short printJob,
    char FAR *queryString
);

```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to modify the SQL query.
queryString	Specifies a pointer to the null-terminated string that you want to use to replace the existing SQL query.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

This function is useful for reports with SQL queries that were explicitly edited in the Show SQL Query dialog box in Seagate Crystal Reports (i.e., those reports that needed database-specific selection criteria or joins). Otherwise it is usually best to continue using function calls such as PESetSelectionFormula and let Seagate Crystal Reports build the SQL query automatically.

PESetSQLQuery has the same restrictions as editing in the Show SQL Query dialog box. In particular, changes are accepted in the WHERE, and ORDER BY clauses but they are ignored in the SELECT list of fields.

NOTE: This call only applies to reports created against an ODBC source or on a native SQL database connection.

PESetSQLQuery (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetSQLQuery Lib "crpe.dll" (ByVal printJob As Integer, ByVal queryString As String) As Integer
```

32-bit

```
Declare Function PESetSQLQuery Lib "crpe32.dll" (ByVal printJob As Integer, ByVal queryString As String) As Integer
```

NOTE: If you are going to include an ORDER BY clause, you must precede it with CHR(13) and CHR(10).

Delphi Syntax

16-bit

```
function PESetSQLQuery (
    printJob: integer;
    queryString: PChar
): Bool;
```

32-bit

```
function PESetSQLQuery (
    printJob: Word;
    queryString: PChar
): Bool stdcall;
```

NOTE: If you are going to include an ORDER BY clause, you must precede it with CHR(13) and CHR(10).

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESetSQLQuery (CWORD, CSTRING) CRPE.DLL
```



PESetTrackCursorInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Sets information for tracking the position of the mouse cursor over the preview window. This function is only valid if the report was sent to a preview window (see *PEOutputToWindow*, Page 257), and if events for the preview window have been enabled (see *PEEnableEvent*, Page 141).

```
BOOL CRPE_API PESetTrackCursorInfo (
    short printJob,
    PETrackCursorInfo FAR *cursorInfo
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you wish to track events.
cursorInfo	Specifies a pointer to <i>PETrackCursorInfo</i> , <i>Page 461</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

- By default, all area's track cursors are arrow cursors. If the preview window contains a drill-down field (database field, summary, group name, group graph), the group area will use a magnify cursor. Group graphs in other areas will also use the magnify cursor.
- This function can be used to give the user visual feed back especially when tracking events.

PESetTrackCursorInfo (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetTrackCursorInfo Lib "crpe.dll" (ByVal  
printJob As Integer, cursorInfo As PETrackCursorInfo) As Integer
```

32-bit

```
Declare Function PESetTrackCursorInfo Lib "crpe.dll" (ByVal  
printJob As Integer, cursorInfo As PETrackCursorInfo) As Integer
```

Delphi Syntax

16-bit

```
function PESetTrackCursorInfo(  
  printJob: Integer;  
  var cursorInfo: PETrackCursorInfo  
): Integer;
```

32-bit

```
function PESetTrackCursorInfo(
    printJob: smallint;
    var cursorInfo: PETrackCursorInfo
): smallint stdcall;
```

PESetWindowOptions

Sets display options for the preview window, including which preview window controls are available. This function must be called after *PEOutputToWindow*, *Page 257*.

Syntax

```
BOOL CRPE_API PESetWindowOptions (
    short printJob
    PEWindowOptions FAR * options
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you wish to set preview window display options.
options	A pointer to a <i>PEWindowOptions</i> , <i>Page 470</i> .

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PESetWindowOptions (Other Declares)

VB Syntax

16-bit

```
Declare Function PESetWindowOptions Lib "crpe.dll" (ByVal
printJob As Integer, options As PEWindowOptions) As Integer
```

32-bit

```
Declare Function PESetWindowOptions Lib "crpe.dll" (ByVal  
printJob As Integer, options As PEWindowOptions) As Integer
```

Delphi Syntax

16-bit

```
function PESetWindowOptions (  
    printJob: Word;  
    var options: PEWindowOptions  
): Bool
```

32-bit

```
function PESetWindowOptions (  
    printJob: Word;  
    var options: PEWindowOptions  
): Bool stdcall;
```

PEShow Functions



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

.Displays the specified page in the preview window. Use these functions any time you want to display specific pages of a report in the preview window or give the user the ability to move forward and backward through a report in the preview window. *PEShowNthPage*, *Page 345*, has some additional considerations and so it is presented in a separate topic.

```
BOOL CRPE_API PEShowNextPage (  
    short printJob  
) ;  
  
BOOL CRPE_API PEShowFirstPage (  
    short printJob  
) ;  
  
BOOL CRPE_API PEShowPreviousPage (  
    short printJob  
) ;  
  
BOOL CRPE_API PEShowLastPage (  
    short printJob  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle for the print job for which you want to indicate the page to be displayed.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEShow Functions (Other Declares)

VB Syntax

16-bit

```
Declare Function PEShowNextPage Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer  
  
Declare Function PEShowFirstPage Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer  
  
Declare Function PEShowPreviousPage Lib "crpe.dll" (ByVal  
printJob As Integer) As Integer  
  
Declare Function PEShowLastPage Lib "crpe.dll" (ByVal printJob  
As Integer) As Integer
```

32-bit

```
Declare Function PEShowFirstPage Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer  
  
Declare Function PEShowLastPage Lib "crpe32.dll" (ByVal printJob  
As Integer) As Integer  
  
Declare Function PEShowNextPage Lib "crpe32.dll" (ByVal printJob  
As Integer) As Integer  
  
Declare Function PEShowPreviousPage Lib "crpe32.dll" (ByVal  
printJob As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEShowNextPage (
    printJob: integer
): Bool;

function PEShowFirstPage (
    printJob: integer
): Bool;

function PEShowPreviousPage (
    printJob: integer
): Bool;

function PEShowLastPage (
    printJob: integer
): Bool;
```

32-bit

```
function PEShowFirstPage (
    printJob: Word
): Bool stdcall;

function PEShowLastPage (
    printJob: Word
): Bool stdcall;

function PEShowNextPage (
    printJob: Word
): Bool stdcall;

function PEShowPreviousPage (
    printJob: Word
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEShowNextPage (CWORD) CRPE.DLL
EXTERN CLOGICAL PEShowFirstPage (CWORD) CRPE.DLL
EXTERN CLOGICAL PEShowPreviousPage (CWORD) CRPE.DLL
EXTERN CLOGICAL PEShowLastPage (CWORD) CRPE.DLL
```

PEShowNthPage



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Displays the specified report page.

```
BOOL CRPE_API PEShowNthPage (
    short printJob,
    short pageN
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job from which you want to display a page.
pageN	Specifies the report page you want to display.

Return value

TRUE (1) if the call is successful; FALSE(0) if an error occurs.

Remarks

Using *PEGetJobStatus*, *Page 171*, you will be able to determine how many pages the given report has. Using this information, you can then ensure that the pageN parameter above does not exceed the total number of pages.

PEShowNthPage (Other Declares)

VB Syntax

16-bit

```
Declare Function PEShowNthPage Lib "crpe.dll" (ByVal printJob As Integer, ByVal pageN As Integer) As Integer
```

32-bit

```
Declare Function PEShowNthPage Lib "crpe32.dll" (ByVal printJob As Integer, ByVal pageN As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEShowNthPage (
    printJob: integer;
    pageN: integer
): Bool;
```

32-bit

```
function PEShowNthPage (
    printJob: Word;
    pageN: Smallint
): Bool stdcall;
```

PEShowPrintControls



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Displays the print controls (the First, Previous, Next, and Last Page buttons as well as the buttons for Cancel, Close, Export, and Print to Printer). Use any time you want to provide control over whether print controls are displayed or not.

```
BOOL CRPE_API PEShowPrintControls (
    short printJob,
    BOOL showPrintControls
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job for which you want to display print controls.
showPrintControls	Indicates whether or not the function is to display print controls. You may use either of the following values:

Value	Meaning
TRUE	The function displays the print controls.
FALSE	The function hides the print controls.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

Print controls are displayed by default. It is not necessary to use this function simply to display controls but only if you want the user to control whether or not the controls are visible.

PEShowPrintControls (Other Declares)

VB Syntax

16-bit

```
Declare Function PEShowPrintControls Lib "crpe.dll" (ByVal  
printJob As Integer, ByVal ShowPrintControls As Integer) As  
Integer
```

32-bit

```
Declare Function PEShowPrintControls Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal ShowPrintControls As Integer) As  
Integer
```

Delphi Syntax

16-bit

```
function PEShowPrintControls (   
    printJob: integer;  
    showPrintControls: Bool  
): Bool;
```

32-bit

```
function PEShowPrintControls(  
    printJob: Word;  
    showPrintControls: Bool  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEShowPrintControls (CWORD, CLOGICAL) CRPE.DLL
```

PStartPrintJob

Starts the printing of a report. This function is used as a mandatory part of each Custom-Print link to trigger the printing of a report to the printer or to the preview window.

```
BOOL CRPE_API PStartPrintJob (
    short printJob,
    BOOL waitUntilDone
);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to start.
waitUntilDone	Indicates if the function is to return as soon as printing starts. Use one of the following values:

Value	Meaning
TRUE	The function returns when printing is finished. In version 1.1 and higher of CRPE, "waitUntilDone" must be true.
FALSE	The function returns as soon as printing starts.

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

Remarks

A print job can be started only once. Once started, the only function that can be used is *PEClosePrintJob*, *Page 129*.

PESStartPrintJob (Other Declares)

VB Syntax

16-bit

```
Declare Function PESStartPrintJob Lib "crpe.dll" (ByVal printJob  
As Integer, ByVal WaitOrNot As Integer) As Integer
```

32-bit

```
Declare Function PESStartPrintJob Lib "crpe32.dll" (ByVal  
printJob As Integer, ByVal WaitOrNot As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PESStartPrintJob (  
    printJob: integer;  
    waitUntilDone: Bool  
): Bool;
```

32-bit

```
function PESStartPrintJob (  
    printJob: Word;  
    waitUntilDone: Bool  
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PESStartPrintJob (CWORD, CLOGICAL) CRPE.DLL
```

PETestNthTableConnectivity

Tests whether a database table's settings are valid and ready to be reported on. This function is typically used if you plan to print at a later time but you want to test now to make sure everything is in order for logging on.

```
BOOL CRPE_API PETestNthTableConnectivity (  
    short printJob,  
    short tableN  
) ;
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job for which you want to test a table's connection settings.
tableN	Specifies the number of the table for which you want to test the connection settings.

Return Value

TRUE (1) if the database session, log on, and location information is all correct; FALSE (0) if something is wrong.

Remarks

This function may require a significant amount of time to complete, since it will first open a user session (if required), then log on to the database server (if required), and then open the appropriate database table (to test that it exists). It does not read any data and closes the table immediately once successful. Logging off is performed when the print job is closed.

- If it fails in any of these steps, the error code set indicates which database information needs to be updated using the functions indicated:
 - If it is unable to begin a session, it sets PE_ERR_DATABASESESSION. The application should update with *PESetNthTableSessionInfo*, *Page 322*.
 - If it is unable to log on to a server, it sets PE_ERR_DATABASELOGON. The application should update with *PESetNthTableLogOnInfo*, *Page 320*.
 - If it is unable to open the table, it sets PE_ERR_DATABASELOCATION. The application should update with *PESetNthTableLocation*, *Page 319*.

PETestNthTableConnectivity (Other Declares)

VB Syntax

16-bit

```
Declare Function PETestNthTableConnectivity Lib "crpe.dll"  
    (ByVal printJob As Integer, ByVal TableN As Integer) As Integer
```

32-bit

```
Declare Function PETestNthTableConnectivity Lib "crpe32.dll" P  
    (ByVal printJob As Integer, ByVal TableN As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PETestNthTableConnectivity (
    printJob,
    tableN: integer;
): Bool;
```

32-bit

```
function PETestNthTableConnectivity (
    printJob: Word;
    tableN: smallint
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PETestNthTableConnectivity (CWORD, CWORD) CRPE.DLL
```

PEZoomPreviewWindow

Changes the magnification of the preview window to a specified level. Use this function when the report has been printed to a preview window and you need to set the magnification of the preview window to a specific level. There are three default magnifications for the preview window: Full Page, Fit One Side, and Fit Both Sides. The magnification level can also be set to a specific percentage.

```
BOOL CRPE_API PEZoomPreviewWindow (
    short printJob,
    short level
);
```

Parameters

Parameter	Description
printJob	The handle of the print job being displayed in the preview window.
level	The zoom level you wish to set the preview window at. This value can be a value from 25 to 400, indicating a magnification percentage, or you can use one of the following constants:

Constant	Value	Meaning
PE_ZOOM_FULL_SIZE	0	Full Page

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_ZOOM_SIZE_FIT_ONE_SIDE	1	Fit One Side
PE_ZOOM_SIZE_FIT_BOTH_SIDES	2	Fit Both Sides

Return Value

TRUE (1) if the call is successful; FALSE (0) if the call fails.

PEZoomPreviewWindow (Other Declares)

VB Syntax

16-bit

```
Declare Function PEZoomPreviewWindow Lib "crpe.dll" (ByVal
printJob As Integer, ByVal ZoomLevel As Integer) As Integer
```

32-bit

```
Declare Function PEZoomPreviewWindow Lib "crpe32.dll" (ByVal
printJob As Integer, ByVal ZoomLevel As Integer) As Integer
```

Delphi Syntax

16-bit

```
function PEZoomPreviewWindow (
  printJob: integer;
  level: integer
): Bool
```

32-bit

```
function PEZoomPreviewWindow (
  printJob: Word;
  level: smallint
): Bool stdcall;
```

dBASE for Windows Syntax

```
EXTERN CLOGICAL PEZoomPreviewWindow (CWORD, CWORD)CRPE.DLL
```

STRUCTURES

COLORREF

The **COLORREF** structure is a 32-bit value that is used for specifying Red, Green, or Blue values. Please refer to the Windows Software Development Kit for complete details on this structure.

DEVMODE

Description

The **DEVMODE** Structure is a Windows API structure that contains information about a printer driver's initialization and environment data. An application passes this structure to the DeviceCapabilities and ExtDeviceMode functions. It is used by the *PESelectPrinter*, *Page 267*, and *PEGetSelectedPrinter*, *Page 222*.

16-bit

```
struct tagDEVMODE
{
    char    dmDeviceName[CCHDEVICENAME];
    UINT    dmSpecVersion;
    UINT    dmDriverVersion;
    UINT    dmSize;
    UINT    dmDriverExtra;
    DWORD   dmFields;
    int     dmOrientation;
    int     dmPaperSize;
    int     dmPaperLength;
    int     dmPaperWidth;
    int     dmScale;
    int     dmCopies;
    int     dmDefaultSource;
    int     dmPrintQuality;
    int     dmColor;
    int     dmDuplex;
    int     dmYResolution;
    int     dmTTOption;
};
```

NOTE: This version of the DEVMODE structure was obtained from Windows 95 SDK.

Members

<i>Member</i>	<i>Description</i>
dmDeviceName	Specifies the name of the device the driver supports. For example, "PCL/HP LaserJet" in the case of the Hewlett-Packard LaserJet. Each driver has a unique string.
dmSpecVersion	Specifies the version number of the DEVMODE structure. For Windows version 3.1, this value should be 0x30A.
dmDriverVersion	Specifies the printer driver version number assigned by the printer driver developer.
dmSize	Specifies the size, in bytes, of the DEVMODE structure. (This value does not include the optional dmDriverData member for device-specific data, which can follow the structure.) If an application manipulates only the driver-independent portion of the data, it can use this member to find out the length of the structure without having to account for different versions.
dmDriverExtra	Specifies the size, in bytes, of the optional dmDriverData member for device-specific data, which can follow the structure. If an application does not use device-specific information, it should set this member to zero.
dmFields	Specifies a set of flags that indicate which of the remaining members in the DEVMODE structure have been initialized. It can be any combination (or it can be none) of the following values:

<i>Constant</i>	<i>Value</i>
DM_ORIENTATION	0x00000001L
DM_PAPERSIZE	0x00000002L
DM_PAPERLENGTH	0x00000004L
DM_PAPERWIDTH	0x00000008L
DM_SCALE	0x00000010L
DM_COPIES	0x0000100L
DM_DEFAULTSOURCE	0x0000200L
DM_PRINTQUALITY	0x0000400L
DM_COLOR	0x0000800L
DM_DUPLEX	0x0001000L
DM_YRESOLUTION	0x0002000L
DM_TTOPTION	0x0004000L

NOTE: A printer driver supports only those members that are appropriate for the printer technology.

<i>Member</i>	<i>Description</i>
dmOrientation	Specifies the orientation of the paper. It can be either DMORIENT_PORTRAIT or DMORIENT_LANDSCAPE.
dmPaperSize	Specifies the size of the paper to print on. This member may be set to zero if the length and width of the paper are specified by the dmPaperLength and dmPaperWidth members, respectively. Otherwise, the dmPaperSize member can be set to one of the following predefined values:

<i>Constant</i>	<i>Description</i>
DMPAPER_FIRST	DMPAPER LETTER
DMPAPER LETTER	Letter, 8 1/2 x 11 in.
DMPAPER LETTERSMALL	Letter Small, 8 1/2 x 11 in.
DMPAPER TABLOID	Tabloid, 11 x 17 in.
DMPAPER LEDGER	Ledger, 17 x 11 in.
DMPAPER LEGAL	Legal, 8 1/2 x 14 in.
DMPAPER STATEMENT	Statement, 5 1/2 x 8 1/2 in.
DMPAPER EXECUTIVE	Executive, 7 1/2 x 10 1/2 in.
DMPAPER A3	A3, 297 x 420 mm
DMPAPER A4	A4, 210 x 297 mm
DMPAPER A4SMALL	A4 Small, 210 x 297 mm
DMPAPER A5	A5, 148 x 210 mm
DMPAPER B4	B4, 250 x 354 mm
DMPAPER B5	B5, 182 x 257 mm
DMPAPER FOLIO	Folio, 8 1/2 x 13 in.
DMPAPER QUARTO	Quarto, 215 x 275 mm
DMPAPER_10X14	10 x 14 in.
DMPAPER_11X17	11 x 17 in.
DMPAPER_NOTE	Note, 8 1/2 x 11 in.
DMPAPER_ENV_9	Envelope #9, 3 7/8 x 8 7/8 in.
DMPAPER_ENV_10	Envelope #10, 4 1/8 x 9 1/2 in.
DMPAPER_ENV_11	Envelope #11, 4 1/2 x 10 3/8 in.
DMPAPER_ENV_12	Envelope #12, 4 1/2 x 11 in.
DMPAPER_ENV_14	Envelope #14, 5 x 11 1/2 in.

<i>Constant</i>	<i>Description</i>
DMPAPER_CSHEET	C size sheet
DMPAPER_DSHEET	D size sheet
DMPAPER_ESHEET	E size sheet
DMPAPER_ENV_DL	Envelope DL, 110 x 220 mm
DMPAPER_ENV_C3	Envelope C3, 324 x 458 mm
DMPAPER_ENV_C4	Envelope C4, 229 x 324 mm
DMPAPER_ENV_C5	Envelope C5, 162 x 229 mm
DMPAPER_ENV_C6	Envelope C6, 114 x 162 mm
DMPAPER_ENV_C65	Envelope C65, 114 x 229 mm
DMPAPER_ENV_B4	Envelope B4, 250 x 353 mm
DMPAPER_ENV_B5	Envelope B5, 176 x 250 mm
DMPAPER_ENV_B6	Envelope B6, 176 x 125 mm
DMPAPER_ENV_ITALY	Envelope, 110 x 230 mm
DMPAPER_ENV_MONARCH	Envelope Monarch, 3 7/8 x 7 1/2 in.
DMPAPER_ENV_PERSONAL	Envelope, 3 5/8 x 6 1/2 in.
DMPAPER_FANFOLD_US	U.S. Standard Fanfold, 14 7/8 x 11 in.
DMPAPER_FANFOLD_STD_GERMAN	German Standard Fanfold, 8 1/2 x 12 in.
DMPAPER_FANFOLD_LGL_GERMAN	German Legal Fanfold, 8 1/2 x 13 in.
DMPAPER_LAST	German Legal Fanfold, 8 1/2 x 13 in.
DMPAPER_USER	User-defined

<i>Member</i>	<i>Description</i>
dmPaperLength	Specifies a paper length, in tenths of a millimeter. This parameter overrides the paper length specified by the dmPaperSize member, either for custom paper sizes or for such devices as dot-matrix printers that can print on a variety of page sizes.
dmPaperWidth	Specifies a paper width, in tenths of a millimeter. This parameter overrides the paper width specified by the dmPaperSize member.

<i>Member</i>	<i>Description</i>
dmScale	Specifies the factor by which the printed output is to be scaled. The apparent page size is scaled from the physical page size by a factor of dmScale/100. For example, a letter-size paper with a dmScale value of 50 would contain as much data as a page of size 17 x 22 inches because the output text and graphics would be half their original height and width.
dmCopies	Specifies the number of copies printed if the device supports multiple-page copies.
dmDefaultSource	Specifies the default bin from which the paper is fed. The application can override this value by using the GETSETPAPERBINS escape. This member can be one of the following values:

<i>Constant</i>
DMBIN_AUTO
DMBIN_LOWER
DMBIN_CASSETTE
DMBIN_MANUAL
DMBIN_ENVELOPE
DMBIN_MIDDLE
DMBIN_ENVMANUAL
DMBIN_ONLYONE
DMBIN_FIRST
DMBIN_SMALLFMT
DMBIN_LARGECAPACITY
DMBIN_TRACTOR
DMBIN_LARGEFORMAT
DMBIN_UPPER
DMBIN_LAST

NOTE: A range of values is reserved for device-specific bins. To be consistent with initialization information, the GETSETPAPERBINS and ENUMPAPERBINS escapes use these values.

<i>Member</i>	<i>Description</i>
dmPrintQuality	Specifies the printer resolution. Following are the four predefined device-independent values:

<i>Constant</i>	<i>Value</i>
DMRES_HIGH	(-4)
DMRES_MEDIUM	(-3)
DMRES_LOW	(-2)
DMRES_DRAFT	(-1)

NOTE: If a positive value is given, it specifies the number of dots per inch (DPI) and is therefore device-dependent.

NOTE: If the printer initializes the dmYResolution member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.

<i>Member</i>	<i>Description</i>								
dmColor	Specifies whether a color printer is to render color or monochrome output. Possible values are:								
	<table border="1"> <thead> <tr> <th><i>Constant</i></th><th><i>Value</i></th></tr> </thead> <tbody> <tr> <td>DMCOLOR_COLOR</td><td>(1)</td></tr> <tr> <td>DMCOLOR_MONOCHROME</td><td>(2)</td></tr> </tbody> </table>	<i>Constant</i>	<i>Value</i>	DMCOLOR_COLOR	(1)	DMCOLOR_MONOCHROME	(2)		
<i>Constant</i>	<i>Value</i>								
DMCOLOR_COLOR	(1)								
DMCOLOR_MONOCHROME	(2)								
<i>Member</i>	<i>Description</i>								
dmDuplex	Specifies duplex (double-sided) printing for printers capable of duplex printing. This member can be one of the following values:								
	<table border="1"> <thead> <tr> <th><i>Constant</i></th><th><i>Value</i></th></tr> </thead> <tbody> <tr> <td>DMDUP_SIMPLEX</td><td>(1)</td></tr> <tr> <td>DMDUP_HORIZONTAL</td><td>(2)</td></tr> <tr> <td>DMDUP_VERTICAL</td><td>(3)</td></tr> </tbody> </table>	<i>Constant</i>	<i>Value</i>	DMDUP_SIMPLEX	(1)	DMDUP_HORIZONTAL	(2)	DMDUP_VERTICAL	(3)
<i>Constant</i>	<i>Value</i>								
DMDUP_SIMPLEX	(1)								
DMDUP_HORIZONTAL	(2)								
DMDUP_VERTICAL	(3)								
<i>Member</i>	<i>Description</i>								
dmYResolution	Specifies the y-resolution of the printer, in dots per inch. If the printer initializes this member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.								
dmTTOption	Specifies how TrueType fonts should be printed. It can be one of the following values:								

<i>Constant</i>	<i>Description</i>
DMTT_BITMAP	Print TrueType fonts as graphics. This is the default action for dot-matrix printers.
DMTT_DOWNLOAD	Download TrueType fonts as soft fonts. This is the default action for Hewlett-Packard printers that use Printer Control Language (PCL).
DMTT_SUBDEV	Substitute device fonts for TrueType fonts. This is the default action for PostScript printers.

Remarks

- Only drivers that are fully updated for Windows versions 3.0 and later, and that export the ExtDeviceMode function use the DEVMODE structure.
- An application can retrieve the paper sizes and names supported by a printer by calling the DeviceCapabilities function with the DC_PAPERS, DC_PAPERSIZE, and DC_PAPERNAME values.
- Before setting the value of the dmTTOption member, applications should find out how a printer driver can use TrueType fonts by calling the DeviceCapabilities function with the DC_TRUETYPE value.
- Drivers can add device-specific data immediately following the DEVMODE structure.

32-bit

```
struct _devicemode
{
    TCHAR          dmDeviceName[ CCHDEVICENAME ] ;
    WORD           dmSpecVersion;
    WORD           dmDriverVersion;
    WORD           dmSize;
    WORD           dmDriverExtra;
    DWORD          dmFields;
    short          dmOrientation;
    short          dmPaperSize;
    short          dmPaperLength;
    short          dmPaperWidth;
    short          dmScale;
    short          dmCopies;
    short          dmDefaultSource;
```

```

short      dmPrintQuality;
short      dmColor;
short      dmDuplex;
short      dmYResolution;
short      dmTTOption;
short      dmCollate;
BCHAR      dmFormName[CCHFORMNAME];
WORD       dmLogPixels;
DWORD      dmBitsPerPel;
DWORD      dmPelsWidth;
DWORD      dmPelsHeight;
DWORD      dmDisplayFlags;
DWORD      dmDisplayFrequency;

#if(WINVER >= 0x0400)
    DWORD      dmICMMethod; // Windows 95 only
    DWORD      dmICMIntent; // Windows 95 only
    DWORD      dmMediaType; // Windows 95 only
    DWORD      dmDitherType; // Windows 95 only
    DWORD      dmReserved1; // Windows 95 only
    DWORD      dmReserved2; // Windows 95 only
#endif /* WINVER >= 0x0400 */
};


```

<i>Member</i>	<i>Description</i>
dmDeviceName	Specifies the name of the device the driver supports. For example, "PCL/HP LaserJet" in the case of the Hewlett-Packard LaserJet. Each driver has a unique string.
dmSpecVersion	Specifies the version number of the DEVMODE structure. For Windows version 3.1, this value should be 0x30A.
dmDriverVersion	Specifies the printer driver version number assigned by the printer driver developer.
dmSize	Specifies the size, in bytes, of the DEVMODE structure. (This value does not include the optional dmDriverData member for device-specific data, which can follow the structure.) If an application manipulates only the driver-independent portion of the data, it can use this member to find out the length of the structure without having to account for different versions.
dmDriverExtra	Specifies the size, in bytes, of the optional dmDriverData member for device-specific data, which can follow the structure. If an application does not use device-specific information, it should set this member to zero.

<i>Member</i>	<i>Description</i>
dmFields	Specifies a set of flags that indicate which of the remaining members in the DEVMODE structure have been initialized. It can be any combination (or it can be none) of the following values:
<i>Constant</i>	<i>Value</i>
DM_ORIENTATION	0x0000001L
DM_PAPERSIZE	0x0000002L
DM_PAPERLENGTH	0x0000004L
DM_PAPERWIDTH	0x0000008L
DM_SCALE	0x00000010L
DM_COPIES	0x0000100L
DM_DEFAULTSOURCE	0x0000200L
DM_PRINTQUALITY	0x0000400L
DM_COLOR	0x0000800L
DM_DUPLEX	0x0001000L
DM_YRESOLUTION	0x0002000L
DM_TTOPTION	0x0004000L

NOTE: A printer driver supports only those members that are appropriate for the printer technology.

<i>Member</i>	<i>Description</i>
dmOrientation	Specifies the orientation of the paper. It can be either DMORIENT_PORTAIT or DMORIENT_LANDSCAPE.
dmPaperSize	Specifies the size of the paper to print on. This member may be set to zero if the length and width of the paper are specified by the dmPaperLength and dmPaperWidth members, respectively. Otherwise, the dmPaperSize member can be set to one of the following predefined values:
<i>Constant</i>	<i>Meaning</i>
DMPAPER_FIRST	DMPAPER LETTER
DMPAPER LETTER	Letter, 8 1/2 x 11 in.
DMPAPER LETTERSMALL	Letter Small, 8 1/2 x 11 in.
DMPAPER_TABLOID	Tabloid, 11 x 17 in.
DMPAPER LEDGER	Ledger, 17 x 11 in.
DMPAPER LEGAL	Legal, 8 1/2 x 14 in.

<i>Constant</i>	<i>Meaning</i>
DMPAPER_STATEMENT	Statement, 5 1/2 x 8 1/2 in.
DMPAPER_EXECUTIVE	Executive, 7 1/2 x 10 1/2 in.
DMPAPER_A3	A3, 297 x 420 mm
DMPAPER_A4	A4, 210 x 297 mm
DMPAPER_A4SMALL	A4 Small, 210 x 297 mm
DMPAPER_A5	A5, 148 x 210 mm
DMPAPER_B4	B4, 250 x 354 mm
DMPAPER_B5	B5, 182 x 257 mm
DMPAPER_FOLIO	Folio, 8 1/2 x 13 in.
DMPAPER_QUARTO	Quarto, 215 x 275 mm
DMPAPER_10X14	10 x 14 in.
DMPAPER_11X17	11 x 17 in.
DMPAPER_NOTE	Note, 8 1/2 x 11 in.
DMPAPER_ENV_9	Envelope #9, 3 7/8 x 8 7/8 in.
DMPAPER_ENV_10	Envelope #10, 4 1/8 x 9 1/2 in.
DMPAPER_ENV_11	Envelope #11, 4 1/2 x 10 3/8 in.
DMPAPER_ENV_12	Envelope #12, 4 1/2 x 11 in.
DMPAPER_ENV_14	Envelope #14, 5 x 11 1/2 in.
DMPAPER_CSHEET	C size sheet
DMPAPER_DSHEET	D size sheet
DMPAPER_ESHEET	E size sheet
DMPAPER_ENV_DL	Envelope DL, 110 x 220 mm
DMPAPER_ENV_C3	Envelope C3, 324 x 458 mm
DMPAPER_ENV_C4	Envelope C4, 229 x 324 mm
DMPAPER_ENV_C5	Envelope C5, 162 x 229 mm
DMPAPER_ENV_C6	Envelope C6, 114 x 162 mm
DMPAPER_ENV_C65	Envelope C65, 114 x 229 mm
DMPAPER_ENV_B4	Envelope B4, 250 x 353 mm
DMPAPER_ENV_B5	Envelope B5, 176 x 250 mm
DMPAPER_ENV_B6	Envelope B6, 176 x 125 mm
DMPAPER_ENV_ITALY	Envelope, 110 x 230 mm
DMPAPER_ENV_MONARCH	Envelope Monarch, 3 7/8 x 7 1/2 in.

<i>Constant</i>	<i>Meaning</i>
DMPAPER_ENV_PERSONAL	Envelope, 3 5/8 x 6 1/2 in.
DMPAPER_FANFOLD_US	U.S. Standard Fanfold, 14 7/8 x 11 in.
DMPAPER_FANFOLD_STD_GERMAN	German Standard Fanfold, 8 1/2 x 12 in.
DMPAPER_FANFOLD_LGL_GERMAN	German Legal Fanfold, 8 1/2 x 13 in.
DMPAPER_LAST	German Legal Fanfold, 8 1/2 x 13 in.
DMPAPER_USER	User-defined

<i>Member</i>	<i>Description</i>
dmPaperLength	Specifies a paper length, in tenths of a millimeter. This parameter overrides the paper length specified by the dmPaperSize member, either for custom paper sizes or for such devices as dot-matrix printers that can print on a variety of page sizes.
dmPaperWidth	Specifies a paper width, in tenths of a millimeter. This parameter overrides the paper width specified by the dmPaperSize member.
dmScale	Specifies the factor by which the printed output is to be scaled. The apparent page size is scaled from the physical page size by a factor of dmScale/100. For example, a letter-size paper with a dmScale value of 50 would contain as much data as a page of size 17 x 22 inches because the output text and graphics would be half their original height and width.
dmCopies	Specifies the number of copies printed if the device supports multiple-page copies.
dmDefaultSource	Specifies the default bin from which the paper is fed.

The application can override this value by using the GETSETPAPERBINS escape. This member can be one of the following values:

<i>Constant</i>
DMBIN_AUTO
DMBIN_LOWER
DMBIN_CASSETTE
DMBIN_MANUAL
DMBIN_ENVELOPE

<i>Constant</i>
DMBIN_MIDDLE
DMBIN_ENVMANUAL
DMBIN_ONLYONE
DMBIN_FIRST
DMBIN_SMALLFMT
DMBIN_LARGEcapacity
DMBIN_TRACTOR
DMBIN_LARGEfmt
DMBIN_UPPER
DMBIN_LAST

NOTE: A range of values is reserved for device-specific bins. To be consistent with initialization information, the GETSETPAPERBINS and ENUMPAPERBINS escapes use these values.

<i>Member</i>	<i>Description</i>
dmPrintQuality	Specifies the printer resolution. Following are the predefined device-independent values:
<i>Constant</i>	
DMRES_DRAFT	(-1)
DMRES_LOW	(-2)
DMRES_MEDIUM	(-3)
DMRES_HIGH	(-4)

NOTE: If a positive value is given, it specifies the number of dots per inch (DPI) and is therefore device-dependent.

NOTE: If the printer initializes the dmYResolution member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.

<i>Member</i>	<i>Description</i>
dmColor	Specifies whether a color printer is to render color or monochrome output. Possible values are:
<i>Constant</i>	
DMCOLOR_COLOR	(1)
DMCOLOR_MONOCHROME	(2)

<i>Member</i>	<i>Description</i>
dmDuplex	Specifies duplex (double-sided) printing for printers capable of duplex printing. This member can be one of the following values:
<i>Constant</i>	<i>Value</i>
DMDUP_SIMPLEX	(1)
DMDUP_HORIZONTAL	(2)
DMDUP_VERTICAL	(3)
<i>Member</i>	<i>Description</i>
dmYResolution	Specifies the y-resolution of the printer, in dots per inch. If the printer initializes this member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.
dmTTOption	Specifies how TrueType fonts should be printed. It can be one of the following values:
<i>Constant</i>	<i>Description</i>
DMTT_BITMAP	Print TrueType fonts as graphics. This is the default action for dot-matrix printers.
DMTT_DOWNLOAD	Download TrueType fonts as soft fonts. This is the default action for Hewlett-Packard printers that use Printer Control Language (PCL).
DMTT_SUBDEV	Substitute device fonts for TrueType fonts. This is the default action for PostScript printers.
<i>Member</i>	<i>Description</i>
dmCollate	Specifies whether collation should be used when printing multiple copies. (This member is ignored unless the printer driver indicates support for collation by setting the dmFields member to DM_COLLATE.) This member can be one of the following values:
<i>Constant</i>	<i>Description</i>
DMCOLLATE_TRUE	Collate when printing multiple copies.

<i>Constant</i>	<i>Description</i>
DMCOLLATE_FALSE	Do not collate when printing multiple copies.

NOTE: Using `DMCOLLATE_TRUE` provides faster, more efficient output for collation, since the data is sent to the device driver just once, no matter how many copies are required. The printer is told to simply print the page again.

<i>Member</i>	<i>Description</i>
dmFormName	Windows NT: Specifies the name of the form to use; for example, "Letter" or "Legal". A complete set of names can be retrieved by using the <code>EnumForms</code> function. Windows 95: Printer drivers do not use this member.
dmLogPixels	Specifies the number of pixels per logical inch.
dmBitsPerPel	Windows NT: Specifies the color resolution, in bits per pixel, of the display device (for example: 4 bits for 16 colors, 8 bits for 256 colors, or 16 bits for 65536 colors). Windows 95: Display drivers use this member, for example, in the <code>ChangeDisplaySettings</code> function. Printer drivers do not use this member.
dmPelsWidth	Windows NT: Specifies the width, in pixels, of the visible device surface. Windows 95: Display drivers use this member, for example, in the <code>ChangeDisplaySettings</code> function. Printer drivers do not use this member.
dmPelsHeight	Windows NT: Specifies the height, in pixels, of the visible device surface. Windows 95: Display drivers use this member, for example, in the <code>ChangeDisplaySettings</code> function. Printer drivers do not use this member.
dmDisplayFlags	Windows NT: Specifies the device's display mode. This member can be one of the following values:

<i>Constant</i>	<i>Description</i>
DM_GRAYSCALE	Specifies that the display is a NON-color device. If this flag is not set, color is assumed.

<i>Constant</i>	<i>Description</i>
DM_INTERLACED	Specifies that the display mode is interlaced. If the flag is not set, NON-interlaced is assumed. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.

<i>Member</i>	<i>Description</i>
dmDisplayFrequency	Windows NT: Specifies the frequency, in hertz (cycles per second), of the display device in a particular mode. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.
dmICMMETHOD	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies how ICM (Image Color Matching) is handled. For a non-ICM application, this member determines if ICM is enabled or disabled. For ICM applications, Windows examines this member to determine how to handle ICM support. This member can be one of the following predefined values, or a driver-defined value greater than the value of DMICMMETHOD_USER:

<i>Constant</i>	<i>Description</i>
DMICMMETHOD_NONE	Windows 95 only: Specifies that ICM is disabled.
DMICMMETHOD_SYSTEM	Windows 95 only: Specifies that ICM is handled by Windows.
DMICMMETHOD_DRIVER	Windows 95 only: Specifies that ICM is handled by the device driver.

<i>Constant</i>	<i>Description</i>
DMICMMETHOD_DEVICE	Windows 95 only: Specifies that ICM is handled by the destination device. The printer driver must provide a user interface for setting this member. Most printer drivers support only the DMICMMETHOD_SYSTEM or DMICMMETHOD_NONE value. Drivers for PostScript printers support all values.

<i>Member</i>	<i>Description</i>
dmICMIntent	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies which of the three possible color matching methods, or intents, should be used by default. This member is primarily for non-ICM applications. ICM applications can establish intents by using the ICM functions. This member can be one of the following predefined values, or a driver defined value greater than the value of DMICM_USER:

<i>Constant</i>	<i>Description</i>
DMICM_SATURATE	Windows 95 only: Color matching should optimize for color saturation. This value is the most appropriate choice for business graphs when dithering is not desired.
DMICM_CONTRAST	Windows 95 only: Color matching should optimize for color contrast. This value is the most appropriate choice for scanned or photographic images when dithering is desired.

<i>Constant</i>	<i>Description</i>
DMICM_COLORMETRIC	Windows 95 only: Color matching should optimize to match the exact color requested. This value is most appropriate for use with business logos or other images when an exact color match is desired.

<i>Member</i>	<i>Description</i>
dmMediaType	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies the type of media being printed on. The member can be one of the following predefined values, or a driver-defined value greater than the value of DMMEDIA_USER:

<i>Constant</i>	<i>Description</i>
DMMEDIA_STANDARD	Windows 95 only: Plain paper.
DMMEDIA_GLOSSY	Windows 95 only: Glossy paper.
DMMEDIA_TRANSPARENCY	Windows 95 only: Transparent film.

<i>Member</i>	<i>Description</i>
dmDitherType	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies how dithering is to be done. The member can be one of the following predefined values, or a driver-defined value greater than the value of DMDITHER_USER:

<i>Constant</i>	<i>Description</i>
DMDITHER_NONE	Windows 95 only: No dithering.
DMDITHER_COARSE	Windows 95 only: Dithering with a coarse brush.
DMDITHER_FINE	Windows 95 only: Dithering with a fine brush.

<i>Constant</i>	<i>Description</i>
DMDITHER_LINEART	Windows 95 only: Line art dithering, a special dithering method that produces well defined borders between black, white, and gray scalings. It is not suitable for images that include continuous graduations in intensity and hue such as scanned photographs.
DMDITHER_GRAYSCALE	Windows 95 only: Device does grayscaling.

<i>Member</i>	<i>Description</i>
dmReserved1	Windows NT: This member is not supported on Windows NT. Windows 95: Not used; must be zero.

<i>Member</i>	<i>Description</i>
dmReserved2	Windows NT: This member is not supported on Windows NT. Windows 95: Not used; must be zero.

Remarks

- A device driver's private data follows the public portion of the DEVMODE structure. The size of the public data can vary for different versions of the structure. The dmSize member specifies the number of bytes of public data, and the dmDriverExtra member specifies the number of bytes of private data.
- For 32-bit, use DEVMODEA.

DEVMODE (Other Declares)

VB Type Listing

This type can not be utilized directly from Visual Basic. Seagate Crystal Reports provides an alternative for Visual Basic developers. Search for crDevMode in Developer's online Help.

Delphi Record Listing

16-bit

The TDevMode record is defined under the WINTYPES.PAS file in Delphi. You do not need to define this record in your own application, simply verify that your application includes WINTYPES.PAS. Documentation of TDevMode is provided here for your own information only.

```
type
  TDevMode = record
    dmDeviceName: array[0..cchDeviceName-1] of Char;
    dmSpecVersion: Word;
    dmDriverVersion: Word;
    dmSize: Word;
    dmDriverExtra: Word;
    dmFields: LongInt;
    dmOrientation: Integer;
    dmPaperSize: Integer;
    dmPaperLength: Integer;
    dmPaperWidth: Integer;
    dmScale: Integer;
    dmCopies: Integer;
    dmDefaultSource: Integer;
    dmPrintQuality: Integer;
    dmColor: Integer;
    dmDuplex: Integer;
  end;
```

<i>Member</i>	<i>Description</i>
dmDeviceName:	Specifies the name of the device the driver supports. For example, "PCL/HP LaserJet" in the case of the Hewlett-Packard LaserJet. Each driver has a unique string.
dmSpecVersion	Specifies the version number of the DEVMODE structure. For Windows version 3.1, this value should be 0x30A.
dmDriverVersion	Specifies the printer driver version number assigned by the printer driver developer.
dmSize	Specifies the size, in bytes, of the DEVMODE structure. (This value does not include the optional dmDriverData member for device-specific data, which can follow the structure.) If an application manipulates only the driver-independent portion of the data, it can use this member to find out the length of the structure without having to account for different versions.

<i>Member</i>	<i>Description</i>
dmDriverExtra	Specifies the size, in bytes, of the optional dmDriverData member for device-specific data, which can follow the structure. If an application does not use device-specific information, it should set this member to zero.
dmFields	Specifies a set of flags that indicate which of the remaining members in the DEVMODE structure have been initialized. It can be any combination (or it can be none) of the following values:

<i>Constant</i>	<i>Value</i>
dm_Orientation	\$0000001
dm_PaperSize	\$0000002
dm_PaperLength	\$0000004
dm_PaperWidth	\$0000008
dm_Scale	\$0000010
dm_Copies	\$0000100
dm_DefaultSource	\$0000200
dm_PrintQuality	\$0000400
dm_Color	\$0000800
dm_Duplex	\$0001000

NOTE: A printer driver supports only those members that are appropriate for the printer technology.

<i>Member</i>	<i>Description</i>
dmOrientation	Specifies the orientation of the paper. It can be either of the following:
<i>Constant</i>	<i>Value</i>
dmorient_Portrait	1
dmorient_Landscape	2

<i>Member</i>	<i>Description</i>
dmPaperSize	Specifies the size of the paper to print on. This member may be set to zero if the length and width of the paper are specified by the dmPaperLength and dmPaperWidth members, respectively. Otherwise, the dmPaperSize member can be set to one of the following predefined values:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
dmpaper_First	1	dmpaper_Letter
dmpaper_Letter	1	Letter 8 1/2 x 11 in
dmpaper_LetterSmall	2	Letter Small 8 1/2 x 11 in
dmpaper_Tabloid	3	Tabloid 11 x 17 in
dmpaper_Ledger	4	Ledger 17 x 11 in
dmpaper_Legal	5	Legal 8 1/2 x 14 in
dmpaper_Statement	6	Statement 5 1/2 x 8 1/2 in
dmpaper_Executive	7	Executive"7 1/2 x 10 in
dmpaper_A3	8	A3 297 x 420 mm
dmpaper_A4	9	A4 210 x 297 mm
dmpaper_A4Small	10	A4 Small 210 x 297 mm
dmpaper_A5	11	A5 148 x 210 mm
dmpaper_B4	12	B4 250 x 354
dmpaper_B5	13	B5 182 x 257 mm
dmpaper_Legal	14	Folio 8 1/2 x 13 in
dmpaper_Quarto	15	Quarto 215 x 275 mm
dmpaper_10X14	16	10x14 in
dmpaper_11X17	17	11x17 in
dmpaper_Note	18	Note 8 1/2 x 11 in
dmpaper_Env_9	19	Envelope #9 3 7/8 x 8 7/8
dmpaper_Env_10	20	Envelope #10 4 1/8 x 9 1/2
dmpaper_Env_11	21	Envelope #11 4 1/2 x 10 3/8
dmpaper_Env_12	22	Envelope #12 4 \276 x 11
dmpaper_Env_14	23	Envelope #14 5 x 11 1/2
dmpaper_CSheet	24	C size sheet
dmpaper_DSheet	25	D size sheet

<i>Constant</i>	<i>Value</i>	<i>Description</i>
dmpaper_ESheet	26	E size sheet
dmpaper_Last	26	dmpaper_ESheet
dmpaper_User	256	user-defined

<i>Member</i>	<i>Description</i>
dmPaperLength	Specifies a paper length, in tenths of a millimeter. This parameter overrides the paper length specified by the dmPaperSize member, either for custom paper sizes or for such devices as dot-matrix printers that can print on a variety of page sizes.
dmPaperWidth	Specifies a paper width, in tenths of a millimeter. This parameter overrides the paper width specified by the dmPaperSize member.
dmScale	Specifies the factor by which the printed output is to be scaled. The apparent page size is scaled from the physical page size by a factor of dmScale/100. For example, a letter-size paper with a dmScale value of 50 would contain as much data as a page of size 17 x 22 inches because the output text and graphics would be half their original height and width.
dmCopies	Specifies the number of copies printed if the device supports multiple-page copies.
dmDefaultSource	Specifies the default bin from which the paper is fed. The application can override this value by using the GETSETPAPERBINS escape. This member can be one of the following values:

<i>Constant</i>	<i>Value</i>
dmbin_First	1
dmbin_Upper	1
dmbin_OnlyOne	1
dmbin_Lower	2
dmbin_Middle	3
dmbin_Manual	4
dmbin_Envelope	5
dmbin_EnvManual	6
dmbin_Auto	7
dmbin_Tractor	8
dmbin_SmallFmt	9

<i>Constant</i>	<i>Value</i>
dmbin_LargeFmt	10
dmbin_LargeCapacity	11
dmbin_Cassette	14
dmbin_Last	dmbin_Cassette
dmbin_User	256

NOTE: A range of values is reserved for device-specific bins. To be consistent with initialization information, the GETSETPAPERBINS and ENUMPAPERBINS escapes use these values.

<i>Member</i>	<i>Description</i>
dmPrintQuality	Specifies the printer resolution. Following are the four predefined device-independent values:
<i>Constant</i>	<i>Value</i>
dmres_Draft	-1
dmres_Low	-2
dmres_Medium	-3
dmres_High	-4

NOTE: If a positive value is given, it specifies the number of dots per inch (DPI) and is therefore device-dependent.

NOTE: If the printer initializes the dmYResolution member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.

<i>Member</i>	<i>Description</i>
dmColor	Specifies whether a color printer is to render color or monochrome output. Possible values are:
<i>Constant</i>	<i>Value</i>
dmcolor_Monochrome	1
dmcolor_Color	2
<i>Member</i>	<i>Description</i>
dmDuplex	Specifies duplex (double-sided) printing for printers capable of duplex printing. This member can be one of the following values:

<i>Constant</i>	<i>Value</i>
dmdup_Simplex	(1)
dmdup_Vertical	(2)
dmdup_Horizontal	(3)

32-bit

```

type
  PDeviceMode = PDeviceModeA;
  dmDeviceName:array[0..CCHDEVICENAME - 1] of AnsiChar;
  dmSpecVersion: Word;
  dmDriverVersion: Word;
  dmSize: Word;
  dmDriverExtra: Word;
  dmFields: DWORD;
  dmOrientation: SHORT;
  dmPaperSize: SHORT;
  dmPaperLength: SHORT;
  dmPaperWidth: SHORT;
  dmScale: SHORT;
  dmCopies: SHORT;
  dmDefaultSource: SHORT;
  dmPrintQuality: SHORT;
  dmColor: SHORT;
  dmDuplex: SHORT;
  dmYResolution: SHORT;
  dmTTOption: SHORT;
  dmCollate: SHORT;
  dmFormName: array[0..CCHFORMNAME - 1] of AnsiChar;
  dmLogPixels: Word;
  dmBitsPerPel: DWORD;
  dmPelsWidth: DWORD;
  dmPelsHeight: DWORD;
  dmDisplayFlags: DWORD;
  dmDisplayFrequency: DWORD;
  dmICMMETHOD: DWORD;
  dmICMIntent: DWORD;
  dmMediaType: DWORD;
  dmDitherType: DWORD;
  dmReserved1: DWORD;
  dmReserved2: DWORD;
end;

```

<i>Member</i>	<i>Description</i>
dmDeviceName	Specifies the name of the device the driver supports. For example, "PCL/HP LaserJet" in the case of the Hewlett-Packard LaserJet. Each driver has a unique string.
dmSpecVersion	Specifies the version number of the DEVMODE structure. For Windows version 3.1, this value should be 0x30A.
dmDriverVersion	Specifies the printer driver version number assigned by the printer driver developer.
dmSize	Specifies the size in bytes, of the DEVMODE structure. (This value does not include the optional dmDriverData member for device-specific data, which can follow the structure.) If an application manipulates only the driver-independent portion of the data, it can use this member to find out the length of the structure without having to account for different versions.
dmDriverExtra	Specifies the size, in bytes, of the optional dmDriverData member for device-specific data, which can follow the structure. If an application does not use device-specific information, it should set this member to zero.
dmFields	Specifies a set of flags that indicate which of the remaining members in the DEVMODE structure have been initialized. It can be any combination (or it can be none) of the following values:

<i>Constant</i>	<i>Value</i>
DM_ORIENTATION	1
DM_PAPERSIZE	2
DM_PAPERLENGTH	4
DM_PAPERWIDTH	8
DM_SCALE	\$10
DM_COPIES	\$100
DM_DEFAULTSOURCE	\$200
DM_PRINTQUALITY	\$400
DM_COLOR	\$800
DM_DUPLEX	\$1000
DM_YRESOLUTION	\$2000
DM_TTOPTION	\$4000
DM_COLLATE	\$8000
DM_FORMNAME	\$10000
DM_LOGPIXELS	\$20000

<i>Constant</i>	<i>Value</i>
DM_BITSPERPEL	\$40000
DM_PELSWIDTH	\$80000
DM_PELSHEIGHT	\$100000
DM_DISPLAYFLAGS	\$200000
DM_DISPLAYFREQUENCY	\$400000
DM_RESERVED1	\$800000
DM_RESERVED2	\$1000000
DM_ICMMETHOD	\$2000000
DM_ICMINTENT	\$4000000
DM_MEDIATYPE	\$8000000
DM_DITHERTYPE	\$10000000

NOTE: A printer driver supports only those members that are appropriate for the printer technology.

<i>Member</i>	<i>Description</i>
dmOrientation	Specifies the orientation of the paper. It can be either of the following:

<i>Constant</i>	<i>Value</i>
DMORIENT_PORTRAIT	1
DMORIENT_LANDSCAPE	2

<i>Member</i>	<i>Description</i>
dmPaperSize	Specifies the size of the paper to print on. This member may be set to zero if the length and width of the paper are specified by the dmPaperLength and dmPaperWidth members, respectively. Otherwise, the dmPaperSize member can be set to one of the following predefined values:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMPAPER LETTER	1	Letter 8 12 x 11 in
DMPAPER FIRST		
DMPAPER LETTERSMALL	2	Letter Small 8 12 x 11 in
DMPAPER TABLOID	3	Tabloid 11 x 17 in
DMPAPER LEDGER	4	Ledger 17 x 11 in

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMPAPER_LEGAL	5	Legal 8 12 x 14 in
DMPAPER_STATEMENT	6	Statement 5 12 x 8 12 in
DMPAPER_EXECUTIVE	7	Executive 7 14 x 10 12 in
DMPAPER_A3	8	A3 297 x 420 mm
DMPAPER_A4	9	A4 210 x 297 mm
DMPAPER_A4SMALL	10	A4 Small 210 x 297 mm
DMPAPER_A5	11	A5 148 x 210 mm
DMPAPER_B4	12	B4 (JIS) 250 x 354
DMPAPER_B5	13	B5 (JIS) 182 x 257 mm
DMPAPER_FOLIO	14	Folio 8 12 x 13 in
DMPAPER_QUARTO	15	Quarto 215 x 275 mm
DMPAPER_10X14	16	10x14 in
DMPAPER_11X17	17	11x17 in
DMPAPER_NOTE	18	Note 8 12 x 11 in
DMPAPER_ENV_9	19	Envelope #9 3 78 x 8 78
DMPAPER_ENV_10	20	Envelope #10 4 18 x 9 12
DMPAPER_ENV_11	21	Envelope #11 4 12 x 10 38
DMPAPER_ENV_12	22	Envelope #12 4 \276 x 11
DMPAPER_ENV_14	23	Envelope #14 5 x 11 12
DMPAPER_CSHEET	24	C size sheet
DMPAPER_DSHEET	25	D size sheet
DMPAPER_ESHEET	26	E size sheet
DMPAPER_ENV_DL	27	Envelope DL 110 x 220 mm
DMPAPER_ENV_C5	28	Envelope C5 162 x 229 mm

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMPAPER_ENV_C3	29	Envelope C3 324 x 458 mm
DMPAPER_ENV_C4	30	Envelope C4 229 x 324 mm
DMPAPER_ENV_C6	31	Envelope C6 114 x 162 mm
DMPAPER_ENV_C65	32	Envelope C65 114 x 229 mm
DMPAPER_ENV_B4	33	Envelope B4 250 x 353 mm
DMPAPER_ENV_B5	34	Envelope B5 176 x 250 mm
DMPAPER_ENV_B6	35	Envelope B6 176 x 125 mm
DMPAPER_ENV_ITALY	36	Envelope 110 x 230 mm
DMPAPER_ENV_MONARCH	37	Envelope Monarch 3.875 x 7.5 in
DMPAPER_ENV_PERSONAL	38	6 34 Envelope 3.58 x 6.12 in
DMPAPER_FANFOLD_US	39	S Std Fanfold 14.78 x 11 in
DMPAPER_FANFOLD_STD_GERMAN	40	German Std Fanfold 8.12 x 12 in
DMPAPER_FANFOLD_LGL_GERMAN	41	German Legal Fanfold 8.12 x 13 in}
DMPAPER_ISO_B4	42	B4 (ISO) 250 x 353 mm
DMPAPER_JAPANESE_POSTCARD	43	Japanese Postcard 100 x 148 mm
DMPAPER_9X11	44	9 x 11 in
DMPAPER_10X11	45	10 x 11 in
DMPAPER_15X11	46	15 x 11 in
DMPAPER_ENV_INVITE	47	Envelope Invite 220 x 220 mm

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMPAPER_RESERVED_48	48	RESERVED - DO NOT USE
DMPAPER_RESERVED_49	49	RESERVED - DO NOT USE
DMPAPER LETTER EXTRA	50	Letter Extra 9 \275 x 12 in
DMPAPER LEGAL EXTRA	51	Legal Extra 9 \275 x 15 in
DMPAPER TABLOID EXTRA	52	Tabloid Extra 11.69 x 18 in
DMPAPER A4 EXTRA	53	A4 Extra 9.27 x 12.69 in
DMPAPER LETTER TRANSVERSE	54	Letter Transverse 8 \275 x 11 in
DMPAPER A4 TRANSVERSE	55	A4 Transverse 210 x 297 mm
DMPAPER LETTER EXTRA TRANSVERSE	56	Letter Extra Transverse 9\275 x 12 in}
DMPAPER A PLUS	57	SuperA-SuperAA4 227 x 356 mm
DMPAPER B PLUS	58	SuperB-SuperBA3 305 x 487 mm
DMPAPER LETTER PLUS	59	Letter Plus 8.5 x 12.69 in
DMPAPER A4 PLUS	60	A4 Plus 210 x 330 mm
DMPAPER A5 TRANSVERSE	61	A5 Transverse 148 x 210 mm
DMPAPER B 5 TRANSVERSE	62	B5 (JIS) Transverse 182 x 257 mm
DMPAPER A3 EXTRA	63	A3 Extra 322 x 445 mm
DMPAPER A5 EXTRA	\$40	A5 Extra 174 x 235 mm
DMPAPER B5 EXTRA	65	B5 (ISO) Extra 201 x 276 mm

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMPAPER_A2	66	A2 420 x 594 mm
DMPAPER_A3 _TRANSVERSE	67	A3 Transverse 297 x 420 mm
DMPAPER_A3_EXTRA_ TRANSVERSE	68	A3 Extra Transverse 322 x 445 mm
DMPAPER_LAST		
DMPAPER_A3_EXTRA_ TRANSVERSE		

<i>Member</i>	<i>Description</i>
dmPaperLength	Specifies a paper length, in tenths of a millimeter. This parameter overrides the paper length specified by the dmPaperSize member, either for custom paper sizes or for such devices as dot-matrix printers that can print on a variety of page sizes.
dmPaperWidth	Specifies a paper width, in tenths of a millimeter. This parameter overrides the paper width specified by the dmPaperSize member.
dmScale	Specifies the factor by which the printed output is to be scaled. The apparent page size is scaled from the physical page size by a factor of dmScale/100. For example, a letter-size paper with a dmScale value of 50 would contain as much data as a page of size 17 by 22 inches because the output text and graphics would be half their original height and width.
dmCopies	Specifies the number of copies printed if the device supports multiple-page copies.
dmDefaultSource	Specifies the default bin from which the paper is fed. The application can override this value by using the GETSETPAPERBINS escape. This member can be one of the following values:

<i>Constant</i>	<i>Value</i>
DMBIN_UPPER	1
DMBIN_FIRST	DMBIN_UPPER
DMBIN_ONLYONE	1
DMBIN_LOWER	2
DMBIN_MIDDLE	3
DMBIN_MANUAL	4

<i>Constant</i>	<i>Value</i>
DMBIN_ENVELOPE	5
DMBIN_ENVMANUAL	6
DMBIN_AUTO	7
DMBIN_TRACTOR	8
DMBIN_SMALLFMT	9
DMBIN_LARGEFORMAT	10
DMBIN_LARGE_CAPACITY	11
DMBIN_CASSETTE	14
DMBIN_FORMSOURCE	15
DMBIN_LAST	DMBIN_FORMSOURCE
DMBIN_USER	\$100 (device specific bins start here)

NOTE: A range of values is reserved for device-specific bins. To be consistent with initialization information, the GETSETPAPERBINS and ENUMPAPERBINS escapes use these values.

<i>Member</i>	<i>Description</i>
dmPrintQuality	Specifies the printer resolution. Following are the four predefined device-independent values:

<i>Constant</i>	<i>Value</i>
DMRES_DRAFT	(-1)
DMRES_LOW	(-2)
DMRES_MEDIUM	(-3)
DMRES_HIGH	(-4)

NOTE: If a positive value is given, it specifies the number of dots per inch (DPI) and is therefore device-dependent.

NOTE: If the printer initializes the dmYResolution member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.

<i>Member</i>	<i>Description</i>
dmColor	Specifies whether a color printer is to render color or monochrome output. Possible values are:

<i>Constant</i>	<i>Value</i>
DMCOLOR_MONOCHROME	1
DMCOLOR_COLOR	2

<i>Member</i>	<i>Description</i>
dmDuplex	Specifies duplex (double-sided) printing for printers capable of duplex printing. This member can be one of the following values:

<i>Constant</i>	<i>Value</i>
DMDUP_SIMPLEX	(1)
DMDUP_VERTICAL	(2)
DMDUP_HORIZONTAL	(3)

<i>Member</i>	<i>Description</i>
dmYResolution	Specifies the y-resolution of the printer, in dots per inch. If the printer initializes this member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.
dmTTOption	Specifies how TrueType fonts should be printed. It can be one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
DMTT_BITMAP	1	Print TT fonts as graphics.
DMTT_DOWNLOAD	2	Download TT fonts as soft fonts.
DMTT_SUBDEV	3	Substitute device fonts for TT fonts.
DMTT_DOWNLOAD_OUTLINE	4	Download TT fonts as outline soft fonts.

<i>Member</i>	<i>Description</i>
dmCollate	Specifies whether collation should be used when printing multiple copies. (This member is ignored unless the printer driver indicates support for collation by setting the dmFields member to DM_COLLATE.) This member can be one of the following values:

<i>Constant</i>	<i>Value</i>
DMCOLLATE_FALSE	0
DMCOLLATE_TRUE	1

NOTE: Using DMCOLLATE_TRUE provides faster, more efficient output for collation, since the data is sent to the device driver just once, no matter how many copies are required. The printer is told to simply print the page again.

<i>Member</i>	<i>Description</i>
dmFormName	Windows NT: Specifies the name of the form to use; for example, "Letter" or "Legal". A complete set of names can be retrieved by using the EnumForms function. Windows 95: Printer drivers do not use this member.
dmLogPixels	Specifies the number of pixels per logical inch.
dmBitsPerPel	Windows NT: Specifies the color resolution, in bits per pixel, of the display device (for example: 4 bits for 16 colors, 8 bits for 256 colors, or 16 bits for 65536 colors). Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.
dmPelsWidth	Windows NT: Specifies the width, in pixels, of the visible device surface. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.
dmPelsHeight	Windows NT: Specifies the height, in pixels, of the visible device surface. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.
dmDisplayFlags	Windows NT: Specifies the device's display mode. This member can be one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DM_GRAYSCALE	1	Specifies that the display is a NON-color device. If this flag is not set, color is assumed.

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DM_INTERLACED	2	Specifies that the display mode is interlaced. If the flag is not set, NON-interlaced is assumed. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.

<i>Member</i>	<i>Description</i>
dmDisplayFrequency	Windows NT: Specifies the frequency, in hertz (cycles per second), of the display device in a particular mode. Windows 95: Display drivers use this member, for example, in the ChangeDisplaySettings function. Printer drivers do not use this member.
dmICMMETHOD	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies how ICM is handled. For a non-ICM application, this member determines if ICM is enabled or disabled. For ICM applications, Windows examines this member to determine how to handle ICM support. This member can be one of the following predefined values, or a driver-defined value greater than the value of DMICMMETHOD_USER:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMICMMETHOD_NONE	1	Windows 95 only: Specifies that ICM is disabled.
DMICMMETHOD_SYSTEM	2	Windows 95 only: Specifies that ICM is handled by Windows.

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMICMMETHOD_DRIVER	3	Windows 95 only: Specifies that ICM is handled by the device driver.
DMICMMETHOD_DEVICE	4	Windows 95 only: Specifies that ICM is handled by the destination device. The printer driver must provide a user interface for setting this member. Most printer drivers support only the DMICMMETHOD _SYSTEM or DMICMMETHOD _NONE value. Drivers for PostScript printers support all values.
DMICMMETHOD_USER	\$100	Device-specific methods start here

<i>Member</i>	<i>Description</i>
dmICMIntent	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies which of the three possible color matching methods, or intents, should be used by default. This member is primarily for non-ICM applications. ICM applications can establish intents by using the ICM functions. This member can be one of the following predefined values, or a driver defined value greater than the value of DMICM_USER:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMICM_SATURATE	1	Maximize color saturation
DMICM_CONTRAST	2	Maximize color contrast

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMICM_COLORMETRIC	3	Use specific color metric
DMICM_USER	\$100	Device-specific intents start here

<i>Member</i>	<i>Description</i>
dmMediaType	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies the type of media being printed on. The member can be one of the following predefined values, or a driver-defined value greater than the value of DMMEDIA_USER:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMMEDIA_STANDARD	1	Standard paper
DMMEDIA_TRANSPARENCY	2	Transparency
DMMEDIA_GLOSSY	3	Glossy paper

<i>Member</i>	<i>Description</i>
dmDitherType	Windows NT: This member is not supported on Windows NT. Windows 95: Specifies how dithering is to be done. The member can be one of the following predefined values, or a driver-defined value greater than the value of DMDITHER_USER:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMDITHER_NONE	1	No dithering
DMDITHER_COARSE	2	Dither with a coarse brush
DMDITHER_FINE	3	Dither with a fine brush
DMDITHER_LINEART	4	LineArt dithering
DMDITHER_ERRORDIFFUSION	5	LineArt dithering
DMDITHER_RESERVED6	6	LineArt dithering
DMDITHER_RESERVED7	7	LineArt dithering
DMDITHER_RESERVED8	8	LineArt dithering
DMDITHER_RESERVED9	9	LineArt dithering

<i>Constant</i>	<i>Value</i>	<i>Description</i>
DMDITHER_GRAYSCALE	10	Device does gray-scaling
DMDITHER_USER	256	Device-specific dithers start here

<i>Member</i>	<i>Description</i>
dmReserved1	Windows NT: This member is not supported on Windows NT. Windows 95: Not used; must be zero.
dmReserved2	Windows NT: This member is not supported on Windows NT. Windows 95: Not used; must be zero.

Remarks

A device driver's private data follows the public portion of the DEVMODE structure. The size of the public data can vary for different versions of the structure. The dmSize member specifies the number of bytes of public data, and the dmDriverExtra member specifies the number of bytes of private data.

PECharSepFileOptions

Description

Contains number and date information used by *PEOutputToFile*, *Page 251*, when you select character separated as your output file format.

NOTE: This structure does not appear in the .H file, .BAS files, .PAS. You must declare it yourself.

VB Type Listing

16-bit

```
Type PECharSepFileOptions
    structSize As Integer
    UseReportNumberFmt As Integer
    UseReportDateFormat As Integer
    StringDelimiter As String
    FieldDelimiter As String * (PE_FIELDDELIMITER)
End Type
```

32-bit

```
Type PECharSepFileOptions
    structSize As Integer
    UseReportNumberFmt As Integer
    reserved1 As Integer
    UseReportDateFormat As Integer
    reserved2 As Integer
    StringDelimiter As String * 1
    FieldDelimiter As String * (PE_FIELDDELIMLEN)
End Type
```

Members

Member	Description
structSize	Specifies the size of the PECharSepFileOptions structure. You must initialize this number to be the size of whatever it is, for example, charSepFileOptions.StructSize = PE_SIZEOF_CHAR_SEP_FILE_OPTIONS.
UseReportNumberFmt	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass TRUE if you want the program to use the same format used in the report, FALSE if you want numbers saved in a format that has been optimized for the file format you have selected.
reserved1	(32-bit only.) Reserved member used internally. Do not set a value.
UseReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass TRUE if you want the program to use the same format as used in the report, FALSE if you want dates saved in a format that has been optimized for the file format you have selected.
reserved2	(32-bit only.) Reserved member used internally. Do not set a value.
StringDelimiter	Specifies the character you want to use to enclose Alphanumeric field data in the character separated values format. You can use whatever character you wish and it must be enclosed in quotes.
FieldDelimiter	Specifies the string you want to use to separate the fields in the character separated value format. Your string may be up to 16 characters long and must be enclosed in quotes.

PECloseButtonClickedEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Description

This function contains close button clicked event information. When the close button in a preview window/preview window is clicked, a callback function will be called with EventId = PE_CLOSE_BUTTON_CLICKED_EVENT. fieldValueList will be released after the callback function. Make a copy of the field value list if you want to hold on to the field value information.

```
struct PECloseButtonClickedEventInfo
{
    WORD StructSize;
    WORD viewIndex;
    long windowHandle;
}
```

Members

Member	Description
StructSize	Specifies the size of the PECloseButtonClickedEventInfo structure. You must initialize this number to be the size of whatever it is, for example, closeButtonClickedEventInfo.StructSize = PE_SIZEOF_CLOSE_BUTTON_CLICKED_EVENT_INFO
viewIndex	Specifies which view is going to be closed. Start from 0.
windowHandle	Specifies the frame window handle the button is on.

PECloseButtonClickedEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PECloseButtonClickedEventInfo
    StructSize As Integer
    viewIndex As Integer
    windowHandle As Long
End Type
```

32-bit

```
Type PECloseButtonClickedEventInfo
    StructSize As Integer
    viewIndex As Integer
    windowHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PECloseButtonClickedEventInfo = record
    StructSize: Word;
    viewIndex: Integer;
    windowHandle: LongInt;
  end;
```

32-bit

```
type
  PECloseButtonClickedEventInfo = record
    StructSize: Word;
    viewIndex: Word;
    windowHandle: HWnd;
  end;
```

PEDrillOnDetailEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides PE_DRILL_ON_DETAIL_EVENT event information.

```
struct PEDrillOnDetailEventInfo
{
    WORD StructSize;
    short selectedFieldIndes;
    long windowHandle;
    struct PEFFieldValueInfo **fieldValueList;
    short nFieldValue;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEDrillOnDetailEventInfo structure. You must initialize this number to be the size of whatever it is, for example,drillOnDetailEventInfo.StructSize = PE_SIZEOF_DRILL_ON_DETAIL_EVENT_INFO.
selectedFieldIndex	A 0 based index indicating which drill-down field was selected. Contains -1 if no field is selected.
windowHandle	Frame window handle where the drill on detail event happens.
fieldValueList	Points to an array of PEFieldValue. Memory pointed by fieldValueList is freed after calling the callback function.
nFieldValue	Number of field value in field value of list (i.e., if the value is listed second, the number would be 2).

Remarks

If the user clicks one of the fields in the Details section, selectedFieldIndex will point to the field index in fieldValueList. These fields have to be one of database field, group name field, summary field, formula field. Clicks on text object, graph, picture, ole, subreport, special var field, or database memo or blob field, selectedFieldIndex return -1.

PEDrillOnDetailEventInfo (Other Declares)

Delphi Record Listing

16-bit

```
type
  PEFieldValueInfoDoublePtr = ^PEFieldValueInfoPtr
  PEDrillOnDetailEventInfo = record{
    StructSize: Word;
    selectedFieldIndes: Integer;
    windowHandle: Longint;
    fieldValueList: PEFieldValueInfoDoublePtr;
    nFieldValue: Integer;
  end;
```

32-bit

```
type
  PEFieldValueInfoDoublePtr = ^PEFieldValueInfoPtr
  PEDrillOnDetailEventInfo = record{
    StructSize: Word;
    selectedFieldIndex: smallint;
    windowHandle: longint;
    fieldValueList: PEFieldValueInfoDoublePtr;
    nFieldValue: smallint;
  end;
```

PEDrillOnGroupEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Description

Specifies drill on group information when PE_DRILL_ON_GROUP_EVENT happens.

```
struct PEDrillOnGroupEventInfo
{
  WORD StructSize;
  WORD drillType;
  long windowHandle;
  char **groupList;
  WORD groupLevel;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEDrillOnGroupEventInfo structure. You must initialize this number to be the size of whatever it is, for example, drillOnGroupEventInfo.StructSize = PE_SIZEOF_DRILL_ON_GROUP_EVENT_INFO.
drillType	Specifies the type of drill down that is used. Use one of the following:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_DE_ON_GROUP	0	Double-click on a group area.
PE_DE_ON_GROUPTREE	1	Double-click a magnifying glass node.
PE_DE_ON_GRAPH	2	Double-click a group graph object.

<i>Member</i>	<i>Description</i>
windowHandle	Points to an array of group name. The list specifies the group path of the selected group. Memory pointed by group list is freed after the callback function.
groupList	Frame window handle where the event happens.
groupLevel	The number of the group name in the group list.

Remarks

groupList will be freed after the callback function. If you want to keep the groupList, make a copy of it.

PEDrillOnGroupEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEDrillOnGroupEventInfo
    StructSize As Integer
    drillType As Integer
    windowHandle As Long
    groupList As String
    groupLevel As Integer
End Type
```

32-bit

```
Type PEDrillOnGroupEventInfo
    StructSize As Integer
    drillType As Integer
    windowHandle As Long
    groupList As String
    groupLevel As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEPCharPointer = ^PChar
  PEDrillOnGroupEventInfo
    StructSize: Integer;
    drillType: Integer;
    windowHandle: Longint;
    groupList: PEPCharPointer;
    groupLevel: Integer;
  end;
```

32-bit

```
type
  PEPCharPointer = ^PChar
  PEDrillOnGroupEventInfo
    StructSize: Word;
    drillType: Word;
    windowHandle: HWnd;
    groupList: PEPCharPointer;
    groupLevel: Word;
  end;
```

PEEnableEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Description

Events are grouped in CRPE. This structure specifies which group event is enabled or disabled.

```
struct PEEnableEventInfo
{
    WORD StructSize;
    short startStopEngine;
    short readingRecordEvent;
    short printWindowButtonEvent;
    short drillEvent;
    short closePrintWindowEvent;
    short activatePrintWindowEvent;
}
```

Members

<i>Member</i>	<i>Description</i>
StructSize	Specifies the size of the PEEnableEventInfo structure. You must initialize this number to be the size of whatever it is, for example, enableEventInfo.StructSize = PE_SIZEOF_ENABLE_EVENT_INFO.
startStopEvent	Indicates a value of PE_START_EVENT, PE_STOP_EVENT, or PE_UNCHANGED for no change.
readingRecordEvent	Indicates a value of PE_READING_RECORDS_EVENT or PE_UNCHANGED for no change.
printWindowButtonEvent	Indicates a value related to the window button, or PE_UNCHANGED for no change.
drillEvent	Indicates a value of PE_DRILL_ON_DETAIL_EVENT, PE_DRILL_ON_GROUP_EVENT, PE_SHOW_GROUP, or PE_UNCHANGED for no change.
closePrintWindowEvent	Indicates a value of PE_CLOSE_PRINT_WINDOW_EVENT, or PE_UNCHANGED for no change.
activatePrintWindowEvent	Indicates a value of PE_ACTIVATE_WINDOW_EVENT, PE_DEACTIVATE_WINDOW_EVENT, or PE_UNCHANGED for no change.

Remarks

- By default, all events are disabled. Use *PEEnableEvent*, *Page 141*, to enable the desired event.
- For startStopEvent, readingRecordEvent, printWindowEvent, drillEvent, closePrintWindowEvent, and activatePrintWindowEvent, use PE_UNCHANGED for no change.

PEEnableEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEEnableEventInfo
    StructSize As Integer
    startStopEngine As Integer
    readingRecordEvent As Integer
    printWindowButtonEvent As Integer
    drillEvent As Integer
    closePrintWindowEvent As Integer
    activatePrintWindowEvent As Integer
End Type
```

32-bit

```
Type PEEnableEventInfo
    StructSize As Integer
    startStopEngine As Integer
    readingRecordEvent As Integer
    printWindowButtonEvent As Integer
    drillEvent As Integer
    closePrintWindowEvent As Integer
    activatePrintWindowEvent As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEEnableEventInfo = record
    StructSize: Word;
    startStopEvent: Integer;
    readingRecordEvent: Integer;
    printWindowButtonEvent: Integer;
    drillEvent: Integer;
    closePrintWindowEvent: Integer;
    activatePrintWindowEvent: Integer;
  end;
```

32-bit

```
type
  PEEnableEventInfo = record
    StructSize: Word;
    startStopEvent: smallint;
    readingRecordEvent: smallint;
    printWindowButtonEvent: smallint;
    drillEvent: smallint;
    closePrintWindowEvent: smallint;
    activatePrintWindowEvent: smallint;
  end;
```

PEExportOptions

Contains file format and output destination information that is used by the *PEGetExportOptions*, *Page 155*, and *PEExportTo*, *Page 146*, functions. The information can be gathered by the *PEGetExportOptions* function and it is used by the *PEExportTo* function for exporting reports.

```
struct PEExportOptions
{
  WORD structSize;
  char formatDLLName [PE_DLL_NAME_LEN];
  DWORD formatType;
  void FAR *formatOptions;
  char destinationDLLName [PE_DLL_NAME_LEN];
  DWORD destinationType;
  void FAR *destinationOptions;
  WORD nFormatOptionsBytes;
  WORD nDestinationOptionsBytes;};
```

Members

<i>Member</i>	<i>Description</i>	
structSize	Specifies the size of the PEExportOptions structure. You must initialize the member to be the size of whatever it is, for example, options.structSize = PE_SIZEOF_EXPORT_OPTIONS.	
formatDLLName	Specifies the pointer to the null-terminated string that contains the format DLL name. The DLL used is determined by the format in which you want to export your report. Select the appropriate DLL name from the table below:	
<i>To export in this format</i>	<i>Use this DLL (16-bit)</i>	<i>Use this DLL (32-bit)</i>
Seagate Crystal Reports Format	uxfcr.dll	u2fcr.dll
Data Interchange Format	uxfwordw.dll	u2fwordw.dll
Word for Windows Format	uxfwordw.dll	u2fwordw.dll
Word for DOS Format (16-bit only)	uxfdoc.dll	u2fdoc.dll
WordPerfect Format (16-bit only)	uxfdoc.dll	u2fdoc.dll
Quattro Pro 5.0 (WB1) Format (16-bit only)	uxfqp.dll	u2fqp.dll
Record Style Format (column of values)	uxfrec.dll	u2frec.dll
Rich Text Format	uxfrtf.dll	u2frtf.dll
Comma Separated Values Format (CSV)	uxfsepv.dll	u2fsepv.dll
Tab Separated Values Format	uxfsepv.dll	u2fsepv.dll
Character Separated Values Format	uxfsepv.dll	u2fsepv.dll

<i>To export in this format</i>	<i>Use this DLL (16-bit)</i>	<i>Use this DLL (32-bit)</i>
Text Format (ASCII)	uxftext.dll	u2ftext.dll
Paginated Text Format (ASCII)	uxftext.dll	u2ftext.dll
Tab Separated Text Format	uxftext.dll	u2ftext.dll
Lotus 1-2-3 (WKS)	uxfwks.dll	u2fwks.dll
Lotus 1-2-3 (WK1)	uxfwks.dll	u2fwks.dll
Lotus 1-2-3 (WK3)	uxfwks.dll	u2fwks.dll
Excel 2.1	uxfxls.dll	u2fxls.dll
Excel 3.0	uxfxls.dll	u2fxls.dll
Excel 4.0	uxfxls.dll	u2fxls.dll
Excel 5.0	uxfxls.dll	u2fxls.dll
Excel 5.0 Tabular		
ODBC	uxfodbc.dll	u2fodbc.dll
HTML	uxfhtml.dll	u2fhtml.dll

<i>Member</i>	<i>Description</i>
formatType	Specifies the type of format you want to use from those types supported by the selected DLL. Whether the format DLL you select supports only one format type (for example, uxfcr.dll) or multiple format types (for example, uxfdoc.dll), you must still fill in this member. Select the format type you want to use from the table below:

<i>To export a report in this format</i>	<i>Use this formatType</i>
Seagate Crystal Reports Format	UXFCrystalReportType
Data Interchange Format	UXFDIFType
Word for Windows Format	UXFWWordWinType
Word for DOS Format	UXFWWordDosType
WordPerfect Format	UXFWordPerfectType
Quattro Pro 5.0 (WB1) Format	UXFQP5Type
Record Style Format (column of values)	UXFRecordType
Rich Text Format	UXFRichTextFormatType

<i>To export a report in this format</i>	<i>Use this formatType</i>
Comma Separated Values Format (CSV)	UXFCommaSeparatedType
Tab Separated Values Format	UXFTabSeparatedType
Character Separated Values Format	UXFCharSeparatedType
Text Format (ASCII)	UXFTextType
Paginated Text Format (ASCII)	UXFPaginatedTextType
Tab Separated Text Format	UXFTabbedTextType
Lotus 1-2-3 (WKS)	UXFLotusWksType
Lotus 1-2-3 (WK1)	UXFLotusWk1Type
Lotus 1-2-3 (WK3)	UXFLotusWk3Type
Excel 2.1	UXFXls2Type
Excel 3.0	UXFXls3Type
Excel 4.0	UXFXls4Type
Excel 5.0	UXFXls5Type
Excel 5.0 Tabular	UXFXlsTypeTab
ODBC	UXFODBCType
HTML	UXFHTML3Type
Microsoft Internet Explorer 2 HTML	UXFExplorer2Type
Netscape 2 HTML	UXFNetscape2Type

<i>Member</i>	<i>Description</i>
formatType1 & 2	Visual Basic (32-bit only)
formatOptions	Specifies a pointer to a structure that supplies date and number information. This information is used by the PEExportOptions structure when you want to export in one of the formats that support date and number options and you want to hard code your options. Select the appropriate structure (if needed) from the table below:

<i>To export a report in this format</i>	<i>Use this structure if you want to hard code formatOptions</i>
Data Interchange Format	<i>UXFDIFOptions, Page 485</i>
Record Style Format (column of values)	<i>UXFRecordStyleOptions, Page 490</i>

<i>To export a report in this format</i>	<i>Use this structure if you want to hard code formatOptions</i>
Comma Separated Values (CSV)	<i>UXFCommaTabSeparatedOptions, Page 484</i>
Tab Separated Values	<i>UXFCommaTabSeparatedOptions, Page 484</i>
Character Separated Values	<i>UXFCharSeparatedOptions, Page 482</i>
Paginated Text	<i>UXFPaginatedTextOptions, Page 489</i>
Excel (Tabular)	
ODBC Format	<i>UXFODBCOptions, Page 488</i>
HTML Format	<i>UXFHTML3Options, Page 487</i>

WARNING: *formatOptions must be pointing to a valid address until PEStartPrintJob, Page 348, is called.*

<i>Member</i>	<i>Description</i>
formatOptions1 & 2	Visual Basic (32-bit only)
destinationDLLName	Specifies a pointer to the null-terminated string that contains the destination DLL name. The DLL used is determined by the destination to which you want to export your report. Select the appropriate DLL name from the table below:

<i>To export a report to this destination</i>	<i>Use this DLL name (16-bit)</i>	<i>Use this DLL name (32-bit)</i>
Disk File	uxddisk.dll	u2ddisk.dll
E-Mail (MAPI)	uxdmapi.dll	u2dmapi.dll
E-Mail (VIM)	uxdvim.dll	u2dvim.dll
Microsoft Exchange	uxdpost.dll	u2dpost.dll

<i>Member</i>	<i>Description</i>
destinationType	Specifies the type of destination you want to use from those types supported by the selected DLL. Even if the destinationDLL name you select supports only one destination type, you must still fill in this member. Select the destination type you want to use from the table below:

<i>To export a report to this destination</i>	<i>Use this destinationType</i>
Disk File	UXDDiskType
E-Mail (MAPI)	UXDMAPIType
E-Mail (VIM)	UXDVIMType
Microsoft Exchange	UXDExchFolderType

<i>Member</i>	<i>Description</i>
destinationType1 & 2	Visual Basic (32-bit only)
destinationOptions	Specifies a pointer to a structure supplying information used by the PEExportOptions structure. This information is needed to export a report and hard code the file name (when exporting to Disk File) or e-mail message information (when exporting to MAPI or VIM destination). Select the appropriate structure (if needed) from the table below:

<i>To export a report to this destination</i>	<i>Use this structure if you want to hard code destinationOptions</i>
Disk File	<i>UXDDiskOptions, Page 475</i>
E-Mail (MAPI)	<i>UXDMAPIOptions, Page 476</i>
E-Mail (VIM)	<i>UXDVIMOptions, Page 481</i>
Microsoft Exchange	<i>UXDPostFolderOptions, Page 479</i>

<i>Member</i>	<i>Description</i>
destinationOptions1 & 2	Visual Basic (32-bit only)
nFormatOptionsBytes	Set by <i>PEGetExportOptions, Page 155</i> , and ignored by <i>PEExportTo, Page 146</i> .
nDestinationOptionsBytes	Set by <i>PEGetExportOptions, Page 155</i> , and ignored by <i>PEExportTo, Page 146</i> .

WARNING: *destinationOptions* must be pointing to a valid address until *PEStartPrintJob, Page 348*, is called.

NOTE: *formatOptions* and *destinationOptions* must be pointing to valid addresses until *PEStartPrintJob* is called.

PEExportOptions (Other Declares)

VB Type Listing

16-bit

```
Type PEExportOptions
    StructSize As Integer
    FormatDLLName As String * PE_DLL_NAME_LEN
    FormatType As Long
    FormatOptions As Long
    DestinationDLLName As String * PE_DLL_NAME_LEN
    DestinationType As Long
    DestinationOptions As Long
    NFormatOptionsBytes As Integer
    NDestinationOptionsBytes As Integer
End Type
```

32-bit

```
Type PEExportOptions
    StructSize As Integer
    FormatDLLName As String * PE_DLL_NAME_LEN
    FormatType1 As Integer
    FormatType2 As Integer
    FormatOptions1 As Integer
    FormatOptions2 As Integer
    DestinationDLLName As String * PE_DLL_NAME_LEN
    DestinationType1 As Integer
    DestinationType2 As Integer
    DestinationOptions1 As Integer
    DestinationOptions2 As Integer
    NFormatOptionsBytes As Integer
    NDestinationOptionsBytes As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEDllNameType = array[0..PE_DLL_NAME_LEN-1] of Char;
  PEExportOptions = record
    StructSize: Word;
    formatDLLName: PEDllNameType;
    formatType: longint;
    formatOptions: Pointer;
    destinationDLLName: PEDllNameType;
    destinationType: longint;
    destinationOptions: Pointer;
    nFormatOptionsBytes: Word;
    nDestinationOptionsBytes: Word;
  end;
```

32-bit

```
type
  PEDllNameType = array[0..PE_DLL_NAME_LEN-1] of Char;
  PEExportOptions = record
    StructSize: Word;
    formatDLLName: PEDllNameType;
    formatType: dWord;
    formatOptions: Pointer;
    destinationDLLName: PEDllNameType;
    destinationType: dWord;
    destinationOptions: Pointer;
    nFormatOptionsBytes: Word;
    nDestinationOptionsBytes: Word;
  end;
```

PEFieldValueInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Specifies a field value in the fieldValueList of *PEDrillOnDetailEventInfo, Page 392*.

```
struct PEFIELDVALUEINFO
{
    WORD StructSize;
    WORD ignored;
    char fieldName [PE_FIELD_NAME_LEN];
    PEValueInfo fieldValue;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEFIELDVALUEINFO structure. You must initialize the member to be the size of whatever it is, for example, fieldValueInfo.structSize = PE_SIZEOF_FIELD_VALUE_INFO
ignored	For 4 byte alignment. Ignore.
fieldName	Specifies the formula form of a field name.
fieldValue	Specifies the field value for the selected Details area for the field specified by the field name.

Remarks

A selected Details area corresponds to a database record.

PEFieldValueInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEFieldValueInfo
    StructSize As Integer
    ignored As Integer
    fieldName As String
    fieldValue As PEValueInfo
End Type
```

32-bit

```
Type PEFieldValueInfo
    StructSize As Integer
    ignored As Integer
    fieldName As String
    fieldValue As PEValueInfo
End Type
```

Delphi Record Listing

16-bit

```
type
  PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of char;
  PEFieldValueInfo = record
    StructSize: Word;
    ignored: Integer;
    fieldName: PChar;
    fieldValue: PEValueInfo;
  end;
```

32-bit

```
type
  PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of char;
  PEFieldValueInfo = record
    StructSize: Word;
    ignored: Word;
    fieldName: PEFieldNameType;
    fieldValue: PEValueInfo;
  end;
```

PEGeneralPrintWindowEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Specifies the general preview window event information.

```
struct PEGeneralPrintWindowEventInfo
{
    WORD StructSize;
    WORD ignored;
    long windowHandle;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEGeneralPrintWindowEventInfo structure. You must initialize the member to be the size of whatever it is, for example, generalPrintWindowEventInfo.structSize = PE_SIZEOF_GENERAL_PRINT_WINDOW_EVENT_INFO.
ignored	For 4 byte alignment. Ignore.
windowHandle	Frame window handle where the event happens.

PEGeneralPrintWindowEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEGeneralPrintWindowEventInfo
    StructSize As Integer
    ignored As Integer
    windowHandle As Long
End Type
```

32-bit

```
Type PEGeneralPrintWindowEventInfo
    StructSize As Integer
    ignored As Integer
    windowHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEGeneralPrintWindowEventInfo = record
    StructSize: Word;
    ignored: Integer;
    windowHandle: Longint;
  end;
```

32-bit

```
type
  PEGeneralPrintWindowEventInfo = record
    StructSize: Word;
    ignored: Word;
    windowHandle: HWnd;
  end;
```

PEGraphDataInfo

Contains information on the report data that is used by a graph. The structure is used by *PEGetGraphData*, *Page 158*, to retrieve current data information and by *PESetGraphData*, *Page 289*, to pass new information.

```
struct PEGraphDataInfo
{
    WORD structSize
    short rowGroupN,
        colGroupN,
        summarizedFieldN,
        graphDirection;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEGraphDataInfo structure. You must initialize this member to be the size of the structure, for example: info.structSize = PE_SIZEOF_GRAPH_DATA_INFO.
rowGroupN	Specifies which group number in the report is used to create the values in the rows of the graph. Use PE_GRAPH_DATA_NULL_SELECTION if you don't want to specify a value.
colGroupN	Specifies which group number in the report is used to create the values in the columns of the graph. Use PE_GRAPH_DATA_NULL_SELECTION if you don't want to specify a value.
summarizedFieldN	Specifies which summary field in the report is used to set the values of the risers in the graph. Summary fields are numbered in order of their creation. Use PE_GRAPH_DATA_NULL_SELECTION if you don't want to specify a value.
graphDirection	For normal group/total report, the direction, is always PE_GRAPH_COLS_ONLY. For cross-tab reports, use any of the following constant values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_GRAPH_ROWS_ONLY	0	Use only row values in graph.
PE_GRAPH_COLS_ONLY	1	Use only column values in graph.
PE_GRAPH_MIXED_ROW_COL	2	Graph by row values, then by column values.
PE_GRAPH_MIXED_COL_ROW	3	Graph by column values, then by row values.
PE_GRAPH_UNKNOWN_DIRECTION	20	The direction of the graph is unknown.

PEGraphDataInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEGraphDataInfo
    structSize As Integer
    RowGroupN As Integer
    ColGroupN As Integer
    SummarizedFieldN As Integer
    GraphDirection As Integer
End Type
```

32-bit

```
Type PEGraphDataInfo
    structSize As Integer
    RowGroupN As Integer
    ColGroupN As Integer
    SummarizedFieldN As Integer
    GraphDirection As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEGraphDataInfo = record*
    StructSize: integer;
    rowGroupN: integer;
    colGroupN: integer;
    summarizedFieldN: integer;
    graphDirection: integer;
  end;
```

32-bit

```
type
  PEGraphDataInfo = record
    StructSize: Word;
    rowGroupN: smallint;
    colGroupN: smallint;
    summarizedFieldN: Word;
    graphDirection: Word;
  end;
```

PEGraphOptions

Contains information on several options available with graphs and charts. This structure is used by *PEGetGraphOptions*, [Page 159](#), to retrieve the current options, and by *PESetGraphOptions*, [Page 291](#), to pass new options.

```
struct PEGraphOptions
{
  WORD structSize;
  double graph.MaxValue,
  double graph.MinValue;
  BOOL showDataValue,
  BOOL showGridLine,
  BOOL verticalBars,
  BOOL showLegend;
  char fontFaceName [PE_GRAPH_TEXT_LEN];
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEGraphOptions structure. You must initialize this member to be the size of the structure, for example: options.structSize = PE_SIZEOF_GRAPH_OPTIONS.
graph.MaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted.
graph.MinValue	Specifies the minimum value that will appear in the graph. Any graph values below this value are not charted.
showDataValue	Specifies whether or not to display the numeric value associated with each riser on the chart. If set to TRUE (1), a value appears in the graph for each riser.
showGridLine	Specifies whether or not to display grid lines on the graph.

<i>Member</i>	<i>Description</i>
verticalBars	Specifies whether to display the bars in a bar graph vertically or horizontally.
showLegend	Specifies whether or not to display the graph legend.
fontFaceName	Specifies the font for all text and values in the entire graph.

PEGraphOptions (Other Declares)

VB Type Listing

16-bit

```
Type PEGraphOptions
    structSize As Integer
    Graph.MaxValue As Double
    Graph.MinValue As Double
    Show.MaxValue As Long
    Show.GridLine As Long
    VerticalBars As Long
    Show.Legend As Long
    FontFaceName As String * PE_GRAPH_TEXT_LEN
End Type
```

32-bit

```
Type PEGraphOptions
    structSize As Integer
    Graph.MaxValue As Double
    Graph.MinValue As Double
    Show.MaxValue As Integer
    Show.GridLine As Integer
    VerticalBars As Integer
    Show.Legend As Integer
    FontFaceName As String * PE_GRAPH_TEXT_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
  PEGraphTextType = array[0..PE_GRAPH_TEXT_LEN-1] of char;
  PEGraphOptions = record
    StructSize: Word;
    graph.MaxValue,
    graph.MinValue: Double;
    showDataValue,
    showGridLine,
    verticalBars,
    showLegend: Bool;
    fontFaceName: PEGraphTextType;
  end;
```

32-bit

```
type
  PEGraphTextType = array[0..PE_GRAPH_TEXT_LEN-1] of char;
  PEGraphOptions = record
    StructSize: Word;
    graph.MaxValue,
    graph.MinValue: Double;
    showDataValue,
    showGridLine,
    verticalBars,
    showLegend: Bool;
    fontFaceName: PEGraphTextType;
  end;
```

PEGraphTextInfo

Contains information about the title and footnote text that appears with a graph. It is used to retrieve current values by *PEGraphText*, *Page 161*, and to pass new values by *PESetGraphText*, *Page 292*.

```
struct PEGraphTextInfo
{
  WORD structSize;
  char graphTitle PE_GRAPH_TEXT_LEN;
  char graphSubTitle PE_GRAPH_TEXT_LEN;
  char graphFootNote PE_GRAPH_TEXT_LEN;
  char graphGroupsTitle PE_GRAPH_TEXT_LEN;
  char graphSeriesTitle PE_GRAPH_TEXT_LEN;
```

```

    char graphXAxisTitle PE_GRAPH_TEXT_LEN;
    char graphYAxisTitle PE_GRAPH_TEXT_LEN;
    char graphZAxisTitle PE_GRAPH_TEXT_LEN;
} ;

```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEGraphTextInfo structure. You must initialize this member to be the size of the structure, for example: info.structSize = PE_SIZEOF_GRAPH_TEXT_INFO.
graphTitle	Specifies the main title text that will appear above your graph.
graphSubTitle	Specifies the sub title text that will appear directly under the main title.
graphFootNote	Specifies the footnote text that will appear under your graph.
graphGroupsTitle	Specifies the title of the groups which are being graphed.
graphSeriesTitle	Specifies the title of the series which is being graphed.
graphXAxisTitle	Specifies the title text that will appear for the X axis. Not valid for Pie graphs.
graphYAxisTitle	Specifies the title text that will appear for the Y axis. Not valid for Pie graphs.
graphZAxisTitle	Specifies the title text that will appear for the Z axis. This value is only valid for 3D graphs.

PEGraphTextInfo (Other Declares)

VB Type Listing

16-bit

```

Type PEGraphTextInfo
    structSize As Integer
    GraphTitle As String * PE_GRAPH_TEXT_LEN
    GraphSubTitle As String * PE_GRAPH_TEXT_LEN
    GraphFootNote As String * PE_GRAPH_TEXT_LEN
    GraphGroupsTitle As String * PE_GRAPH_TEXT_LEN
    GraphSeriesTitle As String * PE_GRAPH_TEXT_LEN
    GraphXAxisTitle As String * PE_GRAPH_TEXT_LEN
    GraphYAxisTitle As String * PE_GRAPH_TEXT_LEN
    GraphZAxisTitle As String * PE_GRAPH_TEXT_LEN
End Type

```

32-bit

```
Type PEGraphTextInfo
    structSize As Integer
    GraphTitle As String * PE_GRAPH_TEXT_LEN
    GraphSubTitle As String * PE_GRAPH_TEXT_LEN
    GraphFootNote As String * PE_GRAPH_TEXT_LEN
    GraphGroupsTitle As String * PE_GRAPH_TEXT_LEN
    GraphSeriesTitle As String * PE_GRAPH_TEXT_LEN
    GraphXAxisTitle As String * PE_GRAPH_TEXT_LEN
    GraphYAxisTitle As String * PE_GRAPH_TEXT_LEN
    GraphZAxisTitle As String * PE_GRAPH_TEXT_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
    PEGraphTextType = array[0..PE_GRAPH_TEXT_LEN-1] of char;
    PEGraphTextInfo = record
        StructSize: Word;
        graphTitle,
        graphSubTitle,
        graphFootNote,
        graphGroupsTitle,
        graphSeriesTitle,
        graphXAxisTitle,
        graphYAxisTitle,
        graphZAxisTitle: PEGraphTextType;
    end;
```

32-bit

```
type
    PEGraphTextType = array[0..PE_GRAPH_TEXT_LEN-1] of char;
    PEGraphTextInfo = record
        StructSize: Word;
        graphTitle,
        graphSubTitle,
        graphFootNote,
        graphGroupsTitle,
        graphSeriesTitle,
        graphXAxisTitle,
        graphYAxisTitle,
        graphZAxisTitle: PEGraphTextType;
    end;
```

PEGroupOptions

```
struct PEGroupOptions
{
    WORD StructSize;
    short condition;
    char fieldName [PE_FIELD_NAME_LEN];
    short sortDirection;
    short repeatGroupHeader;
    short keepGroupTogether;
    short topOrBottomNGroups;
    char topOrBottomNSortFieldName [PE_FIELD_NAME_LEN];
    short nTopOrBottomGroups;
    short discardOtherGroups;
};
```

Members

Member	Description
StructSize	Specifies the size of the PEGroupOptions structure. You must initialize the member to be the size of whatever it is, for example, groupOptions.structSize = PE_SIZEOF_GROUP_OPTIONS. When setting, pass a PE_GC_constant, or PE_UNCHANGED for no change. When getting, use PE_GC_TYPEMASK and PE_GC_CONDITIONMASK to decode the condition.
condition	Specifies a pointer to the condition setting for the selected group section (date and Boolean fields only). See <i>PESetGroupCondition</i> , Page 296, for information on the condition constants.
fieldName	Specifies the field name of the group field.
sortDirection	Specifies one of the following sort directions (or PE_UNCHANGED for no change):

Constant	Value	Meaning
PE_SF_DESCENDING	0	Sorts data in descending order (Z to A, 9 to 1).
PE_SF_ASCENDING	1	Sorts data in ascending order (A to Z, 1 to 9).

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_SF_ORIGINAL	2	Sorts data in its original order. (Group condition only.)
PE_SF_SPECIFIED	3	Sorts data in a specified order. (Group condition only.) Read only.

<i>Member</i>	<i>Description</i>
reoeatGroupHeader:	BOOL value, or PE_UNCHANGED for no change.
keepGoupTogether	BOOL value, or PE_UNCHANGED for no change.
topOrBottomNGroups	Use one of the following values a PE_GO_TBN_constant, or PE_UNCHANGED for no change.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_GO_TBN_ALL_GROUPS _UNSORTED	0	There is no group sorting or Top/ Bottom N for this level of grouping.
PE_GO_TBN_ALL_GROUPS _SORTED	1	There is group sorting, but not Top/Bottom N.
PE_GO_TBN_TOP_N _GROUPS	2	Top N groups will be selected.
PE_GO_TBN_BOTTOM_N _GROUPS	3	Bottom N groups will be selected.

<i>Member</i>	<i>Description</i>
topOrBottomNSortField-Name	Name of the summary field that the groups are ordered by. Remains empty for no change.
nTopOrBottomGroups	Number of groups to select.
discardOtherGroups	Determines whether the remaining groups are discarded or collected into an Others group.

Remarks

- Use PE_UNCHANGED for no change.
- If topOrBottomNGrous is PE_GO_TBN_TOP_N_GROUPS or PE_GO_TBN_BOTTOM_N_GROUPS, all the group sort fields related to this group will be

deleted. A new group sort field will be added with the sort direction of descending or ascending. The group sort field will be sorted by specifying topOrBottomNSortFieldName if it is not empty. It will be sorted by the first group sort field name related to this group (before deleted) if topOrBottomNSortFieldName is empty.

PEGroupOptions (Other Declares)

VB Type Listing

16-bit

```
Type PEGroupOptions
    StructSize As Integer
    ignored As Integer
    windowHandle As Long
End Type
```

32-bit

```
Type PEGroupOptions
    StructSize As Integer
    ignored As Integer
    windowHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of char;
  PEgroupOptions = record
    StructSize: Word;
    condition: smallint;
    fieldName: PEFieldNameType;
    sortDirection: smallint;
    repeatGroupHeader: smallint;
    keepgroupTogether: smallint;
    topOrBottomNGroups: smallint;
    topOrBottomNSortFieldName: smallint;
    nTopOrBottomGroups: smallint;
    discardOtherGroups: smallint
  end;
```

32-bit

```
type
  PEFieldNameType = array[0..PE_FIELD_NAME_LEN-1] of char;
  PEGroupOptions = record
    StructSize: Word;
    condition: smallint;
    fieldName: PEFieldNameType;
    sortDirection: smallint;
    repeatGroupHeader: smallint;
    keepgroupTogether: smallint;
    topOrBottomNGroups: smallint;
    topOrBottomNSortFieldName: PEFieldNameType;
    nTopOrBottomGroups: smallint;
    discardOtherGroups: smallint
  end;
```

PEGroupTreeButtonClickedEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides the group tree button clicked event information when the callback function is called with the event ID equal to PE_GROUP_TREE_BUTTON_CLICKED_EVENT.

```
struct PEGroupTreeButtonClickedEventInfo
{
  WORD StructSize;
  short visible;
  long windowHandle;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEGroupTreeButtonClickedEventInfo structure. You must initialize this member to be the size of the structure, for example: groupTreeButton.StructSize = PE_SIZEOF_GROUP_TREE_BUTTON_CLICKED_EVENT_INFO.
visible	Indicates whether the group tree is shown or hidden.
windowHandle	Frame window handle where the event happens.

PEGroupTreeButtonClickedEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEGroupTreeButtonClickedEventInfo
    StructSize As Integer
    visible As Integer
    windowHandle As Long
End Type
```

32-bit

```
Type PEGroupTreeButtonClickedEventInfo
    StructSize As Integer
    visible As Integer
    windowHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEGroupTreeButtonClickedEventInfo = record
    StructSize: Word;
    visible: Integer;
    windowHandle: Longint;
  end;
```

32-bit

```
type
  PEGroupTreeButtonClickedEventInfo = record
    StructSize: Word;
    visible: smallint;
    windowHandle: HWnd;
  end;
```

PEJobInfo

Retrieves print job process and display information data that is used by *PEGetJobStatus*, *Page 171*.

```
struct PEJobInfo
{
    WORD structSize;
    DWORD NumRecordsRead,
        NumRecordsSelected,
        NumRecordsPrinted;
    WORD DisplayPageN,
        LatestPageN,
        StartPageN;
    BOOL printEnded;
};
```

Members

<i>Member</i>	<i>Description</i>
StructSize	Specifies the size of the PEJobInfo structure. You must initialize this member to be the size of whatever it is, for example, job.StructSize = PE_SIZEOF_JOB_INFO.
NumRecordsRead	Specifies the number of records actually processed.
NumRecordsSelected	Specifies the number of records selected for inclusion in the report out of the total number of records read.
NumRecordsPrinted	Specifies the number of records actually printed or previewed.
DisplayPageN	Specifies the page number of the currently displayed page in the preview window.
LatestPageN	Specifies the largest page number available. Once the printing is complete, this value is the number of the last page.
StartPageN	Specifies the number of the starting page. The value will normally be 1, but you can specify something else using <i>PESetPrintOptions</i> , <i>Page 326</i> .
printEnded	Specifies whether or not the printing process is completed. TRUE indicates that this process is completed; FALSE indicates that is not yet complete. When printing to a preview window, printEnded is True only when the last page is reached.

PEJobInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEJobInfo
    StructSize As Integer
    NumRecordsRead As Long
    NumRecordsSelected As Long
    NumRecordsPrinted As Long
    DisplayPageN As Integer
    LatestPageN As Integer
    StartPageN As Integer
    PrintEnded As Long
End Type
```

32-bit

```
Type PEJobInfo
    StructSize As Integer
    NumRecordsRead As Long
    NumRecordsSelected As Long
    NumRecordsPrinted As Long
    DisplayPageN As Integer
    LatestPageN As Integer
    StartPageN As Integer
    PrintEnded As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEJobInfo = record
    structSize: Word;
    NumRecordsRead: longint;
    NumRecordsSelected: longint;
    NumRecordsPrinted: longint;
    DisplayPageN: Word;
    LatestPageN: Word;
    StartPageN: Word;
    PrintEnded: Bool;
  end;
```

32-bit

```
type
  PEJobInfo = record
    StructSize: Word;
    NumRecordsSelected: longint;
    NumRecordsPrinted: longint;
    DisplayPageN: Word;
    LatestPageN: Word;
    StartPageN: Word;
    PrintEnded: Bool;
  end;
```

PELogOnInfo

Contains log on information that is used by *PEGetNthTableLogOnInfo*, *Page 205*, *PESetNthTableLogOnInfo*, *Page 320*, *PELogOnServer*, *Page 241*, and *PELogOffServer*, *Page 240*, for logging on and off SQL and password-protected non-SQL databases.

```
struct PELogOnInfo
{
  WORD structSize;
  char ServerName [PE_SERVERNAME_LEN];
  char DatabaseName [PE_DATABASENAME_LEN];
  char UserID [PE_USERID_LEN];
  char Password [PE_PASSWORD_LEN];
};
```

Members

<i>Member</i>	<i>Description</i>
StructSize	Specifies the size of the PELogOnInfo structure. You must initialize this member to the size of whatever it is, for example, info.StructSize = PE_SIZEOF_LOGON_INFO.
ServerName	Specifies the logon name for the server used to create the report. *For ODBC, use the data source name. For Essbase and Microsoft Access, see Remarks below.
DatabaseName	Specifies the logon name for the database used to create the report. *
UserIDn	Specifies the user I.D.# necessary to log on to the server. *

<i>Member</i>	<i>Description</i>
Password	Specifies the password necessary to log on to the server. When using this structure to retrieve information using the PEGetNthTableLogOnInfo function, the password parameter is undefined.

*When you pass an empty string ("") for this parameter, the program uses the value that's already set in the report. If you want to override a value that's already set in the report, use a non-empty string (i.e., "Server A"). All strings must be null-terminated.

Remarks

- For Netware SQL, pass the dictionary path name in *ServerName* and data path name in *DatabaseName*.
- If your report uses a Microsoft Access database via ODBC, the data source indicated in the *ServerName* parameter must specify an Access database file. An ODBC data source based on the Access driver with no database specified can not be used at runtime.
- For Essbase databases, pass the Essbase application and database to the *DatabaseName* member with a comma between each. For example:

```
Sample,Basic
```

PELogOnInfo (Other Declares)

VB Type Listing

16-bit

```
Type PELogOnInfo
    structSize As Integer
    ServerName As String * PE_SERVERNAME_LEN
    DatabaseName As String * PE_DATABASENAME_LEN
    UserID As String * PE_USERID_LEN
    Password As String * PE_PASSWORD_LEN
End Type
```

32-bit

```
Type PELogOnInfo
    structSize As Integer
    ServerName As String * PE_SERVERNAME_LEN
    DatabaseName As String * PE_DATABASENAME_LEN
    UserID As String * PE_USERID_LEN
```

```
    Password As String * PE_PASSWORD_LEN  
End Type
```

Delphi Record Listing

16-bit

```
type  
  PELogonServerType = array[0..PE_SERVERNAME_LEN-1] of char;  
  PELogonDBType = array[0..PE_DATABASENAME_LEN-1] of char;  
  PELogonUserType = array[0..PE_USERID_LEN-1] of char;  
  PELogonPassType = array[0..PE_PASSWORD_LEN-1] of char;  
  PELogOnInfo = record  
    structSize: Word;  
    ServerName: PELogonServerType;  
    DatabaseName: PELogonDbType;  
    UserId: PELogonUserType;  
    Password: PELogonPassType;  
  end;
```

32-bit

```
type  
  PELogonServerType = array[0..PE_SERVERNAME_LEN-1] of char;  
  PELogonDBType = array[0..PE_DATABASENAME_LEN-1] of char;  
  PELogonUserType = array[0..PE_USERID_LEN-1] of char;  
  PELogonPassType = array[0..PE_PASSWORD_LEN-1] of char;  
  PELogOnInfo = record  
    StructSize: Word;  
    ServerName: PELogonServerType;  
    DatabaseName: PELogonDbType;  
    UserId: PELogonUserType;  
    Password: PELogonPassType;  
  end;
```

PEParameterFieldInfo

Contains information that is used by the *PEGetNthParameterField*, [Page 197](#), to retrieve current information and *PESetNthParameterField*, [Page 315](#), to pass new information.

```
struct PEParameterFieldInfo  
{  
  WORD structSize;  
  WORD ValueType;  
  WORD DefaultValueSet;
```

```

WORD CurrentValue;
WORD CurrentValueSet;
char Name [PE_PF_NAME_LEN];
char Prompt [PE_PF_PROMPT_LEN];
char DefaultValue [PE_PF_VALUE_LEN];
char CurrentValue [PE_PF_VALUE_LEN];
char ReportName [PE_PF_REPORT_NAME_LEN];
WORD needsCurrentValue
} ;

```

Members (when used with PEGetNthParameterField)

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEParameterFieldInfo structure. You must initialize this member to the size of whatever it is, for example, info.structSize = PE_SIZEOF_PARAMETER_FIELD_INFO.
ValueType	Specifies the data type of the parameter field. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, and string. Use the appropriate constant from the list below.

<i>Constant</i>	<i>Value</i>	<i>Data Type</i>
PE_PF_NUMBER	0	Number
PE_PF_CURRENCY	1	Currency
PE_PF_BOOLEAN	2	Boolean
PE_PF_DATE	3	Date
PE_PF_STRING	4	String

<i>Member</i>	<i>Description</i>
DefaultValueSet	Indicates whether or not a default value was set for the parameter field. The value can be either TRUE (1) if the field was given a default value or FALSE (0) if it was not.
CurrentValueSet	Indicates whether or not a current value was set for the parameter field. A current value can be set through <i>PESetNthParameterField</i> , <i>Page 315</i> , or by a prompting dialog box if the report has saved data.
Name	Specifies the name of the parameter field assigned by the user when the parameter was inserted into the report or the name that was set from code.

<i>Member</i>	<i>Description</i>
Prompt	Specifies the prompting text, if any, that appears in the prompting dialog box. This will be either the prompt assigned by the user when the parameter field was inserted into the report or the prompt that was set from code.
DefaultValue	Specifies the default value assigned to the parameter field if one was set. If the DefaultValueSet member is false, this value is meaningless.
CurrentValue	Specifies the current value assigned to the parameter field. If CurrentValueSet is False, this value is meaningless. To discard the current value, refresh data.
ReportName	The name of the report this parameter field applies to.
needsCurrentValue	TRUE (1) if a current value is required. FALSE (0) if a current value has already been supplied for the parameter field.

Members (when used with PESetNthParameterField)

With PESetNthParameterField, there are two primary uses:

- To set the default value, prompt, and the name.
- To set the current value.

Settings for both alternatives are specified below:

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEParameterFieldInfo structure. You must initialize this member to the size of whatever it is, for example, info.structSize = PE_SIZEOF_PARAMETER_FIELD_INFO.
ValueType	Specifies the data type of the parameter field. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, and string. Use the appropriate constant from the list below.

<i>Data Type</i>	<i>Constant</i>
Number	PE_PF_NUMBER
Currency	PE_PF_CURRENCY
Boolean	PE_PF_BOOLEAN
Date	PE_PF_DATE
String	PE_PF_STRING

<i>Member</i>	<i>Description</i>
DefaultValueSet	When setting the default value from code, set this value to 1 (TRUE).
CurrentValueSet	When setting the current value so the parameter does not prompt, set this value to 1 (TRUE).
Name	When setting the name from code, set this to the name you wish to appear on the tab of the dialog box.
Prompt	When setting the prompt from code, set this to the prompt you want the user to see.
DefaultValue	When setting the default value from code, set the default value you want the user to be prompted with.
CurrentValue	When setting the current value so the parameter does not prompt, set this to the value you want the parameter to use in the report. If CurrentValueSet is False, this item is ignored.
ReportName	The name of the report this parameter field applies to.
needsCurrentValue	TRUE (1) if a current value is required. FALSE (0) if a current value has already been supplied for the parameter field.

Remarks

Strings must be null terminated.

PEParameterFieldInfo (Other Declares)

VB type listing

16-bit

```
Type PEParameterFieldInfo
    structSize As Integer
    valueType As Integer
    DefaultValueSet As Integer
    CurrentValueSet As Integer
    Name As String * PE_PF_NAME_LEN
    Prompt As String * PE_PF_PROMPT_LEN
    DefaultValue As String * PE_PF_VALUE_LEN
    CurrentValue As String * PE_PF_VALUE_LEN
    ReportName As String * PE_PF_REPORT_NAME_LEN
    needsCurrentValue As Integer

End Type
```

32-bit

```
Type PEParameterFieldInfo
    structSize As Integer
    ValueType As Integer
    DefaultValueSet As Integer
    CurrentValueSet As Integer
    Name As String * PE_PF_NAME_LEN
    Prompt As String * PE_PF_PROMPT_LEN
    DefaultValue As String * PE_PF_VALUE_LEN
    CurrentValue As String * PE_PF_VALUE_LEN
    ReportName As String * PE_PF_REPORT_NAME_LEN
    needsCurrentValue As Integer

End Type
```

Delphi listing

16-bit

```
type
  PEParameterFieldValueType
  = array[0..PE_PF_NAME_LEN-1] of char;
  PE_PF_ReportNameType
  = array[0..PE_PF_REPORT_NAME_LEN-1] of char;
  PEParameterFieldInfo = record
    structSize: Word;
    ValueType: Word;
    DefaultValueSet: Word;
    CurrentValueSet: Word;
    Name: PEPromptVarNameType;
    Prompt: PEPromptVarTextType;
    DefaultValue: PEPromptVarValueType;
    CurrentValue: PEPromptVarValueType;
    ReportName: PE_PF_ReportNameType;
    needsCurrentValue: Word
  end;
```

32-bit

```
type
  PEParameterFieldValueType
  = array[0..PE_PF_NAME_LEN-1] of char;
  PE_PF_ReportNameType
  = array[0..PE_PF_REPORT_NAME_LEN-1] of char;
  PEParameterFieldInfo = record
```

```

    structSize: Word;
    ValueType: Word;
    DefaultValueSet: Word;
    CurrentValueSet: Word;
    Name: PEParameterFieldNameType;
    Prompt: PEParameterFieldTextType;
    DefaultValue: PEParameterFieldValueType;
    CurrentValue: PEParameterFieldValueType;
    ReportName: PE_PF_ReportNameType;
    needsCurrentvalue: Word;
end;

```

PEParameterInfo

Contains information that is used by the *PEGetNthParamInfo*, *Page 199*, to gather information about an existing stored procedure parameter.

NOTE: Use this function/structure when you are working with stored procedure parameters, not with parameter fields.

```

struct PEParameterInfo
{
    WORD structSize;
    WORD Type;
    char Name [ PE_PARAMETER_NAME_LEN ];
} ;

```

Members

Member	Description
structSize	Specifies the size of the PEParameterInfo structure. You must initialize this member to be the size of whatever it is, for example, info.structSize = PE_SIZEOF_PARAMETER_INFO.
Type	Specifies the data type of the parameter. The Crystal Report Engine supports the following SQL data types from the list below:

Constant	Value
PE_PT_LONGVARCHAR	-1
PE_PT_BINARY	-2
PE_PT_VARBINARY	-3
PE_PT_LONGVVARBINARY	-4

<i>Constant</i>	<i>Value</i>
PE_PT_BIGINT	-5
PE_PT_TINYINT	-6
PE_PT_BIT	-7
PE_PT_CHAR	1
PE_PT_NUMERIC	2
PE_PT_DECIMAL	3
PE_PT_INTEGER	4
PE_PT_SMALLINT	5
PE_PT_FLOAT	6
PE_PT_REAL	7
PE_PT_DOUBLE	8
PE_PT_DATE	9
PE_PT_TIME	10
PE_PT_TIMESTAMP	11
PE_PT_VARCHAR	12

<i>Member</i>	<i>Description</i>
Name	Specifies the name of the parameter in a null-terminated string.

PEParameterInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEParameterInfo = record
    structSize: Word {initialize to PE_SIZEOF_PARAMETER_INFO.}
    paramType: Word;
    Name:PEProcParamNameType
end;

(*Functions and Procedures*)

(***(Open, print and close report (used when no changes needed to
report-Complete Default))
```

32-bit

```
Type PEParameterInfo
    structSize As Integer
    Type As Integer
    Name As String * PE_PARAMETER_NAME_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
  PEProcParamNameType
  = array[0..PE_PARAMETER_NAME_LEN-1] of char;
  PEParameterInfo = record
    structSize: Word;
    paramType: Word;
    Name: PEProcParamNameType;
  end;
```

32-bit

```
type
  PEProcParamNameType
  = array[0..PE_PARAMETER_NAME_LEN-1] of char;
  PEParameterInfo = record
    structSize: Word;
    paramType: Word;
    Name: PEProcParamNameType;
  end;
```

PEPrintFileOptions

Description

Contains number and date information used by *PEOutputToPrinter*, *Page 254*, when you select any output file format other than Character Separated.

NOTE: This structure does not appear in the .H file, .BAS files, .PAS. You must declare it yourself.

```

struct PEPrintFileOptions
{
    WORD StructSize;
    BOOL UseReportNumberFmt,
        UseReportDateFormat;
};

```

Parameters

<i>Parameter</i>	<i>Description</i>
StructSize	Specifies the size of the structure. You must initialize this member to be the size of whatever it is, for example, print file.StructSize = PE_SIZEOF_PRINT_FILE_OPTIONS.
UseReportNumberFmt	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass True (1) if you want the program to use the same format used in the report, False (0) if you want numbers saved in a format that has been optimized for the file format you have selected.
UseReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass True (1) if you want the program to use the same format as used in the report, False (0) if you want dates saved in a format that has been optimized for the file format you have selected.

Remarks

This functions should be used only for backward compatibility with earlier versions of Seagate Crystal Reports. For new applications, use *PEPrintOptions*, Page 435.

PEPrintOptions

Contains printing specifications that are used by the *PEGetPrintOptions*, Page 211, to retrieve current options and *PESetPrintOptions*, Page 326, to pass new options. These specifications are the same as those that can be set using the Print common dialog box.

```

struct PEPrintOptions
{
    WORD structSize;
    unsigned short startPageN,
                stopPageN;
    unsigned short nReportCopies;

```

```

        unsigned short collation;
};


```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEPrintOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = PE_SIZEOF_PRINT_OPTIONS.
startPageN	Specifies the first page that you want to print. Page numbers are 1 origin (Page 1 = 1, Page 2 = 2, etc.). Enter 0 if you want to retain the existing settings.
stopPageN	Specifies the last page that you want to print. Page numbers are 1 origin (Page 1 = 1, Page 2 = 2, etc.). Enter 0 if you want to retain the existing settings.
nReportCopies	Specifies the number of copies you want to print. Copy numbers, like page numbers, are 1 origin. Enter 0 if you want to retain the existing settings.
collation	Indicates whether or not you want the program to collate the copies (if you are printing multiple copies of a multiple page report). For this parameter, use one of the following constants:

<i>Constant</i>	<i>Value</i>	<i>Action</i>
PE_UNCOLLATED	0	Prints multiple copies of a multiple page report uncollated (Page order = 1, 1, 1, 2, 2, 2, 3, 3, 3, etc.).
PE_COLLATED	1	Prints multiple copies of a multiple page report collated (Page order = 1, 2, 3,..., 1, 2, 3,..., etc.).
PE_DEFAULTCOLLATION	2	Prints multiple copies of a multiple page report using the collation settings as specified in the report.

PEPrintOptions (Other Declares)

VB Type Listing

16-bit

```
Type PEPrintOptions
    structSize As Integer
    StartPageN As Integer
    stopPageN As Integer
    nReportCopies As Integer
    collation As Integer
End Type
```

32-bit

```
Type PEPrintOptions
    structSize As Integer
    StartPageN As Integer
    stopPageN As Integer
    nReportCopies As Integer
    collation As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEPrintOptions = record
    StructSize: Word;
    StartPageN: Word;
    StopPageN: Word;
    nReportCopies: Word;
    Collation: Word;
  end;
```

32-bit

```
type
  PEPrintOptions = record
    StructSize: Word;
    StartPageN: Word;
    StopPageN: Word;
```

```

nReportCopies: Word;
Collation: Word;
end;

```

PEReadingRecordsEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides read records information when a callback function is called with event ID equal to PE_READING_RECORDS_EVENT.

```

struct PEReadingRecordsEventInfo
{
    WORD StructSize;
    short cancelled;
    long recordsRead;
    long recordsSelected;
    short done;
}

```

Members

Member	Description
StructSize	Specifies the size of the PEReadingRecordsEventInfo structure. You must initialize this member to be the size of whatever it is, for example, readingRecords.StructSize = PE_SIZEOF_READING_RECORDS_EVENT_INFO.
cancelled	Indicates whether the reading records is cancelled.
recordsRead	Indicates how many records have been read so far.
recordsSelected	Indicates how many records have been selected so far.
done	Indicates whether reading records is finished.

Remarks

PE_READING_RECORDS_EVENT is not fired for every record. It is fired at the same interval of time the status bar in the Seagate Crystal Report Designer is refreshed for reading record status.

PEReadingRecordsEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEReadingRecordsEventInfo
    StructSize As Integer
    cancelled As Integer
    recordsRead As Long
    recordsSelected As Long
    done As Integer
End Type
```

32-bit

```
Type PEReadingRecordsEventInfo
    StructSize As Integer
    cancelled As Integer
    recordsRead As Long
    recordsSelected As Long
    done As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEReadingRecordsEventInfo = record
    StructSize: Word;
    cancelled: Integer;
    recordsRead: Longint;
    recordsSelected: Longint;
    done: Integer;
  end;
```

32-bit

```
type
  PEReadingRecordsEventInfo = record
    StructSize: Word;
    cancelled: smallint;
    recordsRead: longint;
```

```

recordsSelected: longint;
done: smallint;
end;

```

PEReportSummaryInfo

Provides report summary information. Report summary information corresponds to the report summary information in the Seagate Crystal Reports Designer.

```

struct PEReportSummaryInfo
{
    WORD StructSize;
    char applicationName[PE_APPLICATION_NAME_LEN];
    char title[PE_TITLE_LEN];
    char subject[PE_SI SUBJECT_LEN];
    char author[PE_SI AUTHOR_LEN];
    char keywords[PE_SI KEYWORDS_LEN];
    char comments[PE_SI COMMENTS_LEN];
    char reportTemplate[PE_SI REPORT_TEMPLATE_LEN];
}

```

Parameters

<i>Parameter</i>	<i>Description</i>
StructSize	Specifies the size of the structure. You must initialize this member to be the size of whatever it is, for example, reportSummary.StructSize = PE_SIZEOF_REPORT_SUMMARY_INFO
applicationName	Specifies an application name for the application using this report.
title	Specifies the title of the current report.
subject	Specifies the subject of the current report.
author	Specifies the author of the current report.
keywords	Specifies the keywords included for the current report.
comments	Specifies any comments for the current report.
reportTemplate	Specifies the report template for the current report.

PEReportSummaryInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEReportSummaryInfo
    StructSize As Integer
    applicationName As String * PE_SI_APPLICATION_NAME_LEN
    title As String * PE_SI_TITLE_LEN
    subject As String * PE_SI_SUBJECT_LEN
    keywords As String * PE_SI_KEYWORDS_LEN
    comments As String * PE_SI_COMMENTS_LEN
    reportTemplate As String * PE_SI_REPORT_TEMPLATE_LEN
End Type
```

32-bit

```
Type PEReportSummaryInfo
    StructSize As Integer
    applicationName As String * PE_SI_APPLICATION_NAME_LEN
    title As String * PE_SI_TITLE_LEN
    subject As String * PE_SI_SUBJECT_LEN
    keywords As String * PE_SI_KEYWORDS_LEN
    comments As String * PE_SI_COMMENTS_LEN
    reportTemplate As String * PE_SI_REPORT_TEMPLATE_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
  PEApplicationNameType
    = array[0..PE_SI_APPLICATION_NAME_LEN-1] of char;
  PETitleType = array[0..PE_SI_TITLE_LEN-1] of char;
  PESubjectType = array[0..PE_SI_SUBJECT_LEN-1] of char;
  PEAuthorType = array[0..PE_SI_AUTHOR_LEN-1] of char;
  PEKeywordsType = array[0..PE_SI_KEYWORDS_LEN-1] of char;

  PECommentsType = array[0..PE_SI_COMMENTS_LEN-1] of char;
  PEReportTemplate
    = array[0..PE_SI_REPORT_TEMPLATE_LEN-1] of char;
```

```

PEReportSummary = record
  StructSize: Integer;
  applicationName: PEApplicationNameType;
  title: PETitleType;
  subject: PESubjectType;
  author: PEAuthorType;
  keywords: PEKeywordsType;
  comments: PEReportTemplate;
end;

```

32-bit

```

type
  PEApplicationNameType
    = array[0..PE_SI_APPLICATION_NAME_LEN-1] of char;
  PETitleType = array[0..PE_SI_TITLE_LEN-1] of char;
  PESubjectType = array[0..PE_SI SUBJECT_LEN-1] of char;
  PEAuthorType = array[0..PE_SI_AUTHOR_LEN-1] of char;
  PEKeywordsType = array[0..PE_SI_KEYWORDS_LEN-1] of char;

  PECommentsType = array[0..PE_SI_COMMENTS_LEN-1] of char;
  PEReportTemplate
    = array[0..PE_SI_REPORT_TEMPLATE_LEN-1] of char;

  PEReportSummary = record
    StructSize: Integer;
    applicationName: PEApplicationNameType;
    title: PETitleType;
    subject: PESubjectType;
    author: PEAuthorType;
    keywords: PEKeywordsType;
    comments: PEReportTemplate;
  end;

```

PESearchButtonClickedEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Returns information when a callback function is called with event ID equal to PE_SEARCH_BUTTON_CLICKED_EVENT.

```

struct PESearchButtonClickedEventInfo
{
    long windowHandle;
    char searchString [PE_SEARCH_STRING_LEN];
    WORD StructSize;
}

```

Members

<i>Member</i>	<i>Description</i>
windowHandle	frame window handle which the button is on.
searchString	Search string in search edit control.
StructSize	Specifies the size of the PESearchButtonClickedEventInfo structure. You must initialize this member to be the size of whatever it is, for example, searchButtonClicked.StructSize = PE_SIZEOF_SEARCH_BUTTON_CLICKED_EVENT_INFO.

PESearchButtonClickedEventInfo (Other Declares)

VB Type Listing

16-bit

```

Type PESEarhButtonClickedEventInfo
    windowHandle As Long
    searchString As String * PE_SEARCH_STRING_LEN
    StructSize As Integer
End Type

```

32-bit

```

Type PESEarhButtonClickedEventInfo
    windowHandle As Long
    searchString As String * PE_SEARCH_STRING_LEN
    StructSize As Integer
End Type

```

Delphi Record Listing

16-bit

```
type
  PESearchStringType
  = array[0..PE_SEARCH_STRING_LEN-1] of char;
  PESearchButtonClickedEventInfo = record
    windowHandle: longint;
    searchString: pchar;
    StructSize: integer;
  end;
```

32-bit

```
type
  PESearchStringType
  = array[0..PE_SEARCH_STRING_LEN-1] of char;
  PESearchButtonClickedEventInfo = record
    windowHandle: longint;
    searchString: PESearchStringType;
    StructSize: Word;
  end;
```

PESectionOptions

Contains specifications for formatting selected report sections. This information is used by the *PEGetSectionFormat*, *Page 217*, and *PEGetAreaFormat*, *Page 147*, to retrieve current settings and *PESetSectionFormat*, *Page 331*, and *PESetAreaFormat*, *Page 269*, to pass new settings.

```
struct PESectionOptions
{
  WORD structSize;
  short visible,
  NewPageBefore,
  NewPageAfter,
  KeepTogether,
  SuppressBlankSection,
  ResetPageNAfter,
  printAtBottomOfPage,
  COLORREF backgroundColor
  short underlaySection;
  short showArea;
  short freeFormPlacement;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PESectionOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = PE_SIZEOF_SECTION_OPTIONS.
visible	Specifies whether or not the selected section is to be visible. Pass TRUE to keep the section visible, FALSE to hide the section.
NewPageBefore	Specifies whether or not the program is to insert a page break before the section is printed. Pass TRUE to insert a page break, FALSE not to insert a page break.
NewPageAfter	Specifies whether or not the program is to insert a page break after the section is printed. Pass TRUE to insert a page break, FALSE not to insert a page break.
KeepTogether	Specifies whether or not the program is to keep the section together, either on the current page (if there is room) or on the next (if not). Pass TRUE to keep the section together, FALSE to allow the program to split the section data from one page to the next if necessary.
SuppressBlankSection	Specifies whether or not the program is to eliminate blank sections from your report. A section must be completely empty before the program suppresses. Pass TRUE to eliminate the blank sections, FALSE to retain them.
resetPageNAfter	Specifies whether or not the program is to reset the page number to one (1) for the following page, after it prints a group total. Pass TRUE to reset the page number, FALSE to retain the standard numbering.
printAtBottomOfPage	Specifies whether or not the program is to cause each group value to print only at the bottom of a page; FALSE to have the values print in their normal position.
backgroundColor	Specifies the RGB color value contained in the <i>COLORREF</i> , Page 353 , structure. Use PE_UNCHANGED_COLOR to preserve the existing color or PE_NO_COLOR for no color.
underlaySection	Indicates whether or not the specified section is to underlay the following section(s).
showArea	Specifies True (1) to show an area, False (0) to hide an area. You can drill down on a hidden area.
freeFormPlacement	Design time flag. If set to True (1), an object can be placed anywhere in a section.

Remarks

- All of the structure members except *backgroundColor* are Boolean values. Use PE_UNCHANGED if you want to preserve the existing settings.
- Not all options are available for all sections or areas. For example, if you select a Page Header or Page Footer section or area, you can not change NewPageBefore or NewPageAfter. The only options that are available for all sections are visible and SuppressBlankSection.
- showArea only applies to area format.
- *backgroundColor*, underlaySection, freeFormPlacement, and SuppressBlankSection do not apply to area format.

PESectionOptions (Other Declares)

VB Type Listing

16-bit

```
Type PESectionOptions
    structSize As Integer
    visible As Integer
    NewPageBefore As Integer
    NewPageAfter As Integer
    KeepTogether As Integer
    SuppressBlankSection As Integer
    resetPageNAfter As Integer
    printAtBottomOfPage As Integer
    backgroundColor As Long
    underlaySection As Integer
    showArea As Integer
    freeFormPlacement As Integer
End Type
```

32-bit

```
Type PESectionOptions
    structSize As Integer
    Visible As Integer
    NewPageBefore As Integer
    NewPageAfter As Integer
    KeepTogether As Integer
    SuppressBlankSection As Integer
    ResetPageNAfter As Integer
```

```
    PrintAtBottomOfPage As Integer
    backgroundColor As Long
    underlaySection As Integer
    showArea As Integer
    freeFormPlacement As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PESectionOptions = record
    StructSize: integer;
    visible: integer;
    newPageBefore: integer;
    newPageAfter: integer;
    keepTogether: integer;
    suppressBlankSection: integer;
    resetPageNAfter: integer;
    printAtBottomOfPage: integer;
    backgroundColor: longint;
    underlaySection: integer;
    showArea: integer;
    freeFormPlacement: integer;
  end;
```

32-bit

```
type
  PESectionOptions = record
    StructSize: Word;
    visible: Smallint;
    newPageBefore: Smallint;
    newPageAfter: Smallint;
    keepTogether: Smallint;
    suppressBlankSection: Smallint;
    resetPageNAfter: Smallint;
    printAtBottomOfPage: Smallint;
    backgroundColor: COLORREF;
    underlaySection: Smallint;
    showArea: Smallint;
    freeFormPlacement: Smallint;
  end;
```

PESessionInfo

Contains information about the current Microsoft Access session. Use this structure with *PEGetNthTableSessionInfo*, *Page 206*, to retrieve current information and *PESetNthTableSessionInfo*, *Page 322*, to pass new information.

```
struct PESessionInfo
{
    WORD structSize;
    char UserID [PE_SESS_PASSWORD_LEN-1];
    char Password [PE_SESS_PASSWORD_LEN-1];
    DWORD SessionHandle;
};
```

Members

Member	Description
structSize	Specifies the size of the PESessionInfo structure. You must initialize this member to be the size of whatever it is, for example location.structSize = PE_SIZEOF_SESSION_INFO. All strings are null-terminated.
UserID	Specifies the user ID value needed for logging on to the system.
Password	Specifies the password needed for logging on to the system. Read only.
SessionHandle	The handle to the current MS Access session.

Remarks

- When using this structure with *PEGetNthTableSessionInfo*, the password member is not defined. You specify the password when using the structure with *PESetNthTableSessionInfo*.
- The SessionHandle member is likewise undefined with the *PEGetNthTableSessionInfo* function. When you're using the *PESetNthTableSessionInfo* structure, if the SessionHandle value is 0, the Crystal Report Engine uses the UserID and Password settings. In all other cases it uses the SessionHandle value.
- In Microsoft Access 95 and later, an Access database can have session security (also known as user-level security), database-level security, or both. If the Access database contains only session security, simply pass the session password to the Password member before calling the *PESetNthTableSessionInfo*, *Page 322*, function. If the Access database contains database-level security, use a newline character, '\n' (ASCII character 10) followed by the database-level password. For example:

```
"\ndbpassword"
```

If the Access database contains both session security and database-level security, use the session password followed by the newline character and the database password:

```
"sesspswd\ndbpassword"
```

Alternately, database-level security can also be handled by assigning the database-level password to the Password member of the *PELogOnInfo*, *Page 425*, structure and calling *PESetNthTableLogOnInfo*, *Page 320*.

PESessionInfo (Other Declares)

VB Type Listing

16-bit

```
Type PESessionInfo
    structSize As Integer
    UserID As String * PE_SESS_USERID_LEN
    Password As String * PE_SESS_PASSWORD_LEN
    SessionHandle As Long
End Type
```

32-bit

```
Type PESessionInfo
    structSize As Integer
    UserID As String * PE_SESS_USERID_LEN
    Password As String * PE_SESS_PASSWORD_LEN
    SessionHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PESesPassType = array[1..PE_SESS_PASSWORD_LEN] of char;
  PESessionInfo = record
    StructSize: Word;
    UserID: PESesPassType;
    Password: PESesPassType;
    SessionHandle: longint;
  end;
```

32-bit

```
type
  PESEsPassType = array[1..PE_SESS_PASSWORD_LEN] of char;
  PESessionInfo = record
    StructSize: Word;
    UserID: PESEsPassType;
    Password: PESEsPassType;
    SessionHandle: DWord;
  end;
```

PEShowGroupEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides show group event information when an event callback is called with event ID equal to PE_SHOW_GROUP_EVENT.

```
struct PEShowGroupEventInfo
{
  WORD StructSize;
  WORD groupLevel;
  long windowHandle;
  char **groupList;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEShowGroupEventInfo structure. You must initialize this member to be the size of whatever it is, for example showGroup.StructSize = PE_SIZEOF_SHOW_GROUP_EVENT_INFO.
groupLevel	Number of items in groupList.
windowHandle	Frame window handle where the event happens.
groupList	List of group names describing the group path that the group area is going to navigate to. This pointer is freed after the callback function.

Remarks

Make a copy of the groupList if you want to keep group path information. CRPE frees the groupList after the callback function.

PEShowGroupEventInfo (Other Declares)

Delphi Record Listing

16-bit

```
type
  PEPCharPointer = ^PChar;
  PEShowGroupEventInfo = record
    StructSize: Word;
    groupLevel: Word;
    windowHandle: longint;
    groupList: PEPCharPointer;
  end;
```

32-bit

```
type
  PEPCharPointer = ^PChar;
  PEShowGroupEventInfo = record
    StructSize: Word;
    groupLevel: Word;
    windowHandle: Longint;
    groupList: PEPCharPointer;
  end;
```

PEStartEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides start event information when the callback function is called with event ID equal to PE_START_EVENT.

```
struct PEStartEventInfo
{
  WORD StructSize;
```

```

    WORD destination;
}

```

Members

<i>Member</i>	<i>Description</i>		
StructSize	Specifies the size of the PEStartEventInfo structure. You must initialize this member to be the size of whatever it is, for example startEvent.StructSize = PE_SIZEOF_START_EVENT_INFO.		
destination	Indicates the process destination as one of the following values:		
	<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
	PE_TO_NOWHERE	0	No destination.
	PE_TO_WINDOW	1	Print to window.
	PE_TO_PRINTER	2	Print to printer.
	PE_TO_EXPORT	3	Export.
	PE_FROM_QUERY	4	From a query.

PEStartEventInfo (Other Declares)

VB Type Listing

16-bit

```

Type PEStartEventInfo
    StructSize As Integer
    destination As Integer
End Type

```

32-bit

```

Type PEStartEventInfo
    StructSize As Integer
    destination As Integer
End Type

```

Delphi Record Listing

16-bit

```
type
  PEStartEventInfo = record
    StructSize: Word;
    destination: Integer;
  end;
```

32-bit

```
type
  PEStartEventInfo = record
    StructSize: Word;
    destination: Word;
  end;
```

PEStopEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Provides stop event information when a callback function is called with an event ID equal to PE_STOP_EVENT.

```
struct PEStopEventInfo
{
  WORD StructSize;
  WORD destination;
  WORD jobStatus;
}
```

Members

Member	Description
StructSize	Specifies the size of the PEStopEventInfo structure. You must initialize this member to be the size of whatever it is, for example stopEvent.StructSize = PE_SIZEOF_STOP_EVENT_INFO.
destination	Indicates the process destination as one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_TO_NOWHERE	0	No destination.
PE_TO_WINDOW	1	Print to window.
PE_TO_PRINTER	2	Print to printer.
PE_TO_EXPORT	3	Export.
PE_FROM_QUERY	4	From a query.

<i>Member</i>	<i>Description</i>
jobStatus	Indicates the current status of the job as one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_JOBNOTSTARTED	1	The job has not been started.
PE_JOBINPROGRESS	2	The job is currently in progress.
PE_JOBCOMPLETED	3	The job has completed successfully.
PE_JOBFAILED	4	The job failed.
PE_JOBCANCELLED	5	The job was cancelled by the user.
PE_JOBHALTED	6	The report has exceeded the record or time governors.

PEStopEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEStopEventInfo
    StructSize As Integer
    destination As Integer
    jobStatus As Integer
End Type
```

32-bit

```
Type PEStopEventInfo
  StructSize As Integer
  destination As Integer
  jobStatus As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEStopEventInfo = record
    StructSize: Word;
    destination: Integer;
    jobStatus: Integer;
  end;
```

32-bit

```
type
  PEStopEventInfo = record
    StructSize: Word;
    destination: Word;
    jobStatus: Word;
  end;
```

PESubreportInfo

Used by the *PEGetSubreportInfo*, *Page 228*, function to gather information about a specified subreport.

```
struct PESubreportInfo
{
  WORD structSize;
  char name [PE_SUBREPORT_NAME_LEN-1];
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PESubreportInfo structure. You must initialize this member to be the size of whatever it is, for example location.structSize = PE_SIZEOF_SUBREPORT_INFO.

<i>Member</i>	<i>Description</i>
name	Specifies the name of the subreport. This is the name assigned to the subreport when it was first created.

PESubreportInfo (Other Declares)

VB Type Listing

16-bit

```
Type PESubreportInfo
    structSize:Word; {initialize to PE_SIZEOF_SUBREPORT_INFO.}
    name: PE_Subreport_Name_Type;
end;

{store Proc Parameter Structures}
```

32-bit

```
Type PESubreportInfo
    structSize As Integer
    Name As String * PE_SUBREPORT_NAME_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
  PESubreportNameType
  = array[0..PE_SUBREPORT_NAME_LEN-1] of char;
  PESubreportInfo = record
    structSize: Word;
    name: PESubreportNameType;
  end;
```

32-bit

```
type
  PESubreportNameType
  = array[0..PE_SUBREPORT_NAME_LEN-1] of char;
  PESubreportInfo = record
    structSize: Word;
    name: PESubreportNameType;
  end;
```

PETableLocation

Contains database location information that is used with the *PEGetNthTableLocation*, *Page 203*, to gather current location information and *PESetNthTableLocation*, *Page 319*, to pass new location information.

```
struct PETableLocation
{
    WORD structSize;
    char Location [PE_SIZEOF_TABLELOCATION];
};
```

Members

Member	Description
structSize	Specifies the size of the PETableLocation structure. You must initialize this member to be the size of whatever it is, for example location.structSize = PE_SIZEOF_TABLE_LOCATION.
Location	Specifies the database location. This member is database dependent and must be formatted correctly for the expected database, for example:

Examples
xBASE (Natively): <drive>:\<path>\<file>
xBASE (ODBC): <datasource name>
Paradox (Natively): <drive>:\<path>\<file>
Paradox (ODBC): <datasource name>
Btrieve (Natively): <drive>:\<path>\<file>
Btrieve (ODBC): <datasource name>
Oracle (Natively): <database>.<table>
Oracle (ODBC): <database>.<table>
SQL Server (Natively): <database>.<owner>.<table>
SQL Server (ODBC): <database>.<owner>.<table>

PETableLocation (Other Declares)

VB Type Listing

16-bit

```
Type PETableLocation
    structSize As Integer
    Location As String * PE_TABLE_LOCATION_LEN
End Type
```

32-bit

```
Type PETableLocation
    structSize As Integer
    Location As String * PE_TABLE_LOCATION_LEN
End Type
```

Delphi Record Listing

16-bit

```
type
  PETableLocType
  = array[0..PE_TABLE_LOCATION_LEN-1] of char;
  PETableLocation = record
    StructSize: Word;
    Location: PETableLocType;
  end;
```

32-bit

```
type
  PETableLocType
  = array[0..PE_TABLE_LOCATION_LEN-1] of char;
  PETableLocation = record
    StructSize: Word;
    Location: PETableLocType;
  end;
```

PETableType

Contains information for identifying the type of a specified table. This information is gathered using [PEGetNthTableType](#), *Page 207*.

```
struct PETableType
{
    WORD structSize;
    char DLLName [PE_DLL_NAME_LEN-1];
    char DescriptiveName [PE_FULL_NAME_LEN-1];
    WORD DBType;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PETableType structure. You must initialize this member to be the size of whatever it is, for example, TableType.structSize = PE_SIZEOF_TABLE_TYPE.
DLLName	Specifies the name of the appropriate database DLL for the table of interest (in quotes). Select the DLL you want to use from the table below:

<i>Use this DLL</i>	<i>For this standard non-SQL database</i>
PDBBDE.DLL	Borland Database Engine
PDBBND.DLL	Bound reports
PDBDAO.DLL	DAO data sources (Access)
PDBJET.DLL	Access
PDBPDX.DLL	Paradox
PDBXBSE.DLL	dBASE, FoxPro, Clipper
PDCTBTRV.DLL	Btrieve

<i>Use this DLL</i>	<i>For this SQL database</i>
PDSDB22.DLL	DB2/2
PDSGUPTA.DLL	Gupta
PDSNETW.DLL	Netware
PDSODBC.DLL	ODBC
PDSORACL.DLL	Oracle
PDSSYB10.DLL	Sybase 10/11

<i>Use this DLL</i>	<i>For this SQL database</i>
PDSSYBAS.DLL	Sybase

<i>Member</i>	<i>Description</i>
DescriptiveName	Specifies the name of the table of interest (in quotes).
DBType	Specifies the type of database that contains the table of interest. Select DB type from the table below:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_DT_STANDARD	1	The type for standard, non-SQL databases.
PE_DT_SQL	2	The type for SQL databases.

PETableType (Other Declares)

VB Type Listing

16-bit

```
Type PETableType
    structSize As Integer
    DLLName As String * PE_DLL_NAME_LEN
    DescriptiveName As String *
        PE_FULL_NAME_LEN
    DBType As Integer
End Type
```

32-bit

```
Type PETableType
    structSize As Integer
    DLLName As String * PE_DLL_NAME_LEN
    DescriptiveName As String *
        PE_FULL_NAME_LEN
    DBType As Integer
End Type
```

Delphi Record Listing

16-bit

```
type
  PEDllNameType = array[0..PE_DLL_NAME_LEN-1] of char;
  PEFullNameType = array[0..PE_FULL_NAME_LEN-1] of char;
  PETableType = record
    StructSize: Word;
    DLLName: PEDllNameType;
    DescriptiveName: PEFullNameType;
    DBType: Word;
  end;
```

32-bit

```
type
  PEDllNameType = array[0..PE_DLL_NAME_LEN-1] of char;
  PEFullNameType = array[0..PE_FULL_NAME_LEN-1] of char;
  PETableType = record
    StructSize: Word;
    DLLName: PEDllNameType;
    DescriptiveName: PEFullNameType;
    DBType: Word;
  end;
```

PETrackCursorInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

This structure is used to get or set the track cursor in the preview window by *PEGetTrackCursorInfo*, *Page 229*, or *PESetTrackCursorInfo*, *Page 339*.

```
struct PETrackCursorInfo
{
  WORD structSize
  short groupAreaCursor;
  short groupAreaFieldCursor;
  short detailAreaCursor;
  short detailAreaFieldCursor;
  short graphCursor;
  long groupAreaCursorHandle;
  long groupAreaFieldCursorHandle;
```

```

    long detailAreaCursorHandle;
    long detailAreaFieldCursorHandle;
    long graphCursorHandle;
}

```

Members

<i>Member</i>	<i>Description</i>
StructSize	Specifies the size of the PETrackCursorInfo structure. You must initialize this member to be the size of whatever it is, for example trackCursor.StructSize = PE_SIZEOF_TRACK_CURSOR_INFO.
groupAreaCursor	Specifies the group area cursor. Pass PE_UNCHANGED for no change, or use one of the following values:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_TC_DEFAULT_CURSOR	0	Sets the default cursor to PE_TC_ARROW_CURSOR.
PE_TC_ARROW_CURSOR	1	Arrow cursor.
PE_TC_CROSS_CURSOR	2	
PE_TC_IBEAM_CURSOR	3	I-beam cursor
PE_TC_UPARROW_CURSOR	4	Arrow cursor pointing to the top of the screen.
PE_TC_SIZEALL_CURSOR	5	32-bit only.
PE_TC_SIZENWSE_CURSOR	6	Sizing cursor when resizing from the top, left-hand side of the screen to the bottom, right-hand side of the screen.
PE_TC_SIZENESW_CURSOR	7	Sizing cursor when resizing from the top, right-hand side of the screen to the bottom, left-hand side of the screen.

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_TC_SIZEWE_CURSOR	8	Sizing cursor when resizing from the left side of the screen to the right side of the screen.
PE_TC_SIZENS_CURSOR	9	Sizing cursor when resizing from the top of the screen to the bottom of the screen.
PE_TC_NO_CURSOR	10	32-bit only.
PE_TC_WAIT_CURSOR	11	Wait (i.e., hourglass) cursor
PE_TC_APPSTARTING_CURSOR	12	32-bit only.
PE_TC_HELP_CURSOR	13	32-bit only.
PE_TC_SIZE_CURSOR	14	16-bit only.
PE_TC_ICON_CURSOR	15	16-bit only.
PE_TC_MAGNIFY_CURSOR	99	Magnifying glass cursor (used for drill-down)

<i>Parameter</i>	<i>Description</i>
groupAreaFieldCursor	Specifies the cursor for a memo, blob, database, summary, or formula field in the group area. See the list of cursor constants under groupAreaCursor.
detailAreaCursor	Specifies the cursor for the Details area. See the list of cursor constants under groupAreaCursor.
deatailAreaFieldCursor	Specifies the cursor for a memo, blob, database, summary, or formula field in the Details area. See the list of cursor constants under groupAreaCursor.
graphCursor	Specifies the cursor for the group graph in the Report Header or Report Footer area. See the list of cursor constants under groupAreaCursor.
groupAreaCursorHandle	Reserved, do not use.
groupAreaFeidlCursor-Handle	Reserved, do not use.

<i>Parameter</i>	<i>Description</i>
detailAreaCursorHandle	Reserved, do not use.
detailAreaFieldCursorHandle	Reserved, do not use.
graphCursorHandle	Reserved, do not use.

Remarks

- By default, all the cursors are PE_TC_ARROW_CURSOR. If the canDrillDown option in *PEWindowOptions*, *Page 470*, is True, groupAreaCursor, groupAreaFieldCursor, and graphCursor will be PE_TC_MAGNIFY_CURSOR.
- Some cursors are 32-bit only and some are 16-bit only. If a cursor is not supported on a platform, the cursor will not be changed for the specified area.

PETrackCursorInfo (Other Declares)

VB Type Listing

16-bit

```
Type PETrackCursorInfo
    groupAreaCursor As Integer
    groupAreaFieldCursor As Integer
    detailAreaCursor As Integer
    detailAreaFieldCursor As Integer
    graphCursor As Integer
    groupAreaCursorHandle As Long
    groupAreaFieldCursorHandle As Long
    detailAreaCursorHandle As Long
    detailAreaFieldCursorHandle As Long
    graphCursorHandle As Long
End Type
```

32-bit

```
Type PETrackCursorInfo
    groupAreaCursor As Integer
    groupAreaFieldCursor As Integer
    detailAreaCursor As Integer
    detailAreaFieldCursor As Integer
    graphCursor As Integer
    groupAreaCursorHandle As Long
    groupAreaFieldCursorHandle As Long
```

```
    detailAreaCursorHandle As Long
    detailAreaFieldCursorHandle As Long
    graphCursorHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PETrackCursorInfo = record
    groupAreaCursor: Integer;
    groupAreaFieldCursor: Integer;
    detailAreaCursor: Integer;
    detailAreaFieldCursor: Integer;
    graphCursor: Integer;
    groupAreaCursorHandle: Longint;
    groupAreaFieldCursorHandle: Longint;
    detailAreaCursorHandle: Longint;
    detailAreaFieldCursorHandle: Longint;
    graphCursorHandle: Longint;
  end;
```

32-bit

```
type
  PETrackCursorInfo = record
    StructSize: Word;
    groupAreaCursor: smallint;
    groupAreaFieldCursor: smallint;
    detailAreaCursor: smallint;
    detailAreaFieldCursor: smallint;
    graphCursor: smallint;
    groupAreaCursorHandle: longint;
    groupAreaFieldCursorHandle: longint;
    detailAreaCursorHandle: longint;
    detailAreaFieldCursorHandle: longint;
    graphCursorHandle: longint; {
  end;
```

PEValueInfo

Contains information that is used by the *PEConvertPFIInfoToVInfo*, *Page 133*, and *PEConvertVInfoToPFIInfo*, *Page 135*, functions. The structure is used by the *PEConvertPFIInfoToVInfo* function to return converted parameter values in simple types and by the *PEConvertVInfoToPFIInfo* function to accept values for conversion to the binary format required by *PESetNthParameterField*, *Page 315*.

```
struct PEValueInfo
{
    WORD StructSize;
    WORD valueType;
    double viNumber;
    double viCurrency;
    BOOL viBoolean;
    char viString[PE_VI_STRING_LEN];
    short viDate[3];
    short viDateTime[6];
    short viTime[3];
    COLORREF viColor;
    short viInteger;
    char viC;
    char ignored;
    long viLong
} PEValueInfo;
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the PEValueInfo structure. You must initialize this member to the size of whatever it is, for example, info.structSize = PE_SIZEOF_VALUE_INFO.
valueType	Specifies the data type of the parameter field. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, and string. Use the appropriate constant from the list below:

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_VI_NUMBER	0	Number
PE_VI_CURRENCY	1	Currency
PE_VI_BOOLEAN	2	Boolean
PE_VI_DATE	3	Date

<i>Constant</i>	<i>Value</i>	<i>Meaning</i>
PE_VI_STRING	4	String
PE_VI_DATETIME	5	DateTime
PE_VI_TIME	6	Time
PE_VI_INTEGER	7	Integer or Short
PE_VI_COLOR	8	COLOREF
PE_VI_CHAR	9	Char
PE_VI_LONG	10	Long
PE_VI_NOVALUE	100	No Value

<i>Member</i>	<i>Description</i>
viNumber	Specifies the value if the parameter is a numeric value.
viCurrency	Specifies the value if the parameter is a currency value.
viBoolean	Specifies the value if the parameter is a Boolean value.
viString	Specifies the value if the parameter is a string value.
viDate	Specifies the value if the parameter is a date value (year, month, day).
viDateTime	For future support of dateTime parameters.
viTime	For future support of time parameters.
viColor	For future support of color parameters.
viInteger	For future support of integer parameters.
viC	For future support of character parameters.
ignored	Internal use only. Do not use.
viLong	For future support of long value.

Remarks

Strings must be null terminated.

PEValueInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEValueInfo
    StructSize As Integer
    valueType As Integer
    viNumber As Double
    viCurrency As Double
    viBoolean As Integer
    viString As String * PE_VI_STRING_LEN
    viDate(0 To 2) As Integer
    viDateTime(0 To 5) As Integer
    viTime(0 To 2) As Integer
    viColor As Long
    viInteger As Integer
    viC As String * 1
    ignored As String
    vilong As Long
End Type
```

32-bit

```
Type PEValueInfo
    StructSize As Integer
    valueType As Integer
    viNumber As Double
    viCurrency As Double
    viBoolean As Long
    viString As String * PE_VI_STRING_LEN
    viDate(0 To 2) As Integer
    viDateTime(0 To 5) As Integer
    viTime(0 To 2) As Integer
    viColor As Long
    viInteger As Integer
    viC As Byte
    ignored As String
    vilong As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEVALUEINFOSTRINGTYPE
  = array[0..PE_VI_STRING_LEN-1] of smallint;
  PEVALUEINFODATEORTIMETYPE = array[0..5] of smallint;
  PEVALUEINFODATETIMETYPE = array[0..2] of smallint;
  PEValueInfo = record
    StructSize: Word;
    valueType: Word;
    viNumber: Double;
    viCurrency: Double;
    viBoolean: BOOL;
    viString: PEVALUEINFOSTRINGTYPE;
    viDate: PEVALUEINFODATEORTIMETYPE;
    viDateTime: PEVALUEINFODATETIMETYPE;
    viTime: PEVALUEINFODATEORTIMETYPE;
    viColor: Longint;
    viInteger: Smallint;
    viC: Char;
    ignored: Char;
    viLong: Longint
  end;
```

32-bit

```
type
  PEVALUEINFOSTRINGTYPE
  = array[0..PE_VI_STRING_LEN-1] of smallint;
  PEVALUEINFODATEORTIMETYPE = array[0..5] of smallint;
  PEVALUEINFODATETIMETYPE = array[0..2] of smallint;
  PEValueInfo = record
    StructSize:Word;
    valueType: Word; {a PE_VI_constant
    viNumber: Double;
    viCurrency: Double;
    viBoolean: BOOL;
    viString: PEVALUEINFOSTRINGTYPE;
    viDate: PEVALUEINFODATEORTIMETYPE;
    viDateTime: PEVALUEINFODATETIMETYPE;
    viTime: PEVALUEINFODATEORTIMETYPE;
    viColor: COLORREF;
    viInteger: Smallint;
    viC: Char; Char;{BYTE}}
```

```

    ignored: Char;
    viLong: Longint;
end;

```

PEWindowOptions

This structure is used by *PEGetWindowOptions*, *Page 234*, and *PESetWindowOptions*, *Page 341*, to get or set preview window contents.

```

struct PEWindowOptions
{
    WORD StructSize;
    short hasGroupTree;
    short canDrillDown;
    short hasNavigationControls;
    short hasCancelButton;
    short hasPrintButton;
    short hasExportButton;
    short hasZoomControl;
    short hasCloseButton;
    short hasProgressControls;
    short hasSearchButton;
    short hasPrintSetupButton;
    short hasRefreshButton
}

```

Members

<i>Value</i>	<i>Meaning</i>
StructSize	Specifies the size of the PEWindowOptions structure. You must initialize this member to the size of whatever it is, for example, info.structSize = PE_SIZEOF_WINDOW_OPTIONS.
hasGroupTree	Specifies whether or not the Group Tree will appear in the window. Whether or not the preview window has a group tree is determined by two flags. One is this option (hasGroupTree) and the other is the hasGroupTreeOption in Seagate Crystal Reports (Report Options dialog box). By default, this value is set to False.
canDrillDown	Specifies whether or not drill-down capabilities will be active in the window. By default, this value is set to False. Use PE_UNCHANGED for no change.

<i>Value</i>	<i>Meaning</i>
hasNavigationControls	Specifies whether or not navigation controls will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasCancelButton	Specifies whether or not a Cancel button will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasPrintButton	Specifies whether or not a Print button will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasExportButton	Specifies whether or not an Export button will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasZoomControl	Specifies whether or not zoom controls will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasCloseButton	Specifies whether or not a Close button will appear in the window. By default, this value is set to False. If canDrillDown is set to True, CRPE will turn hasCloseButton on unless you set it to False.
hasProgressControls	Specifies whether or not progress controls will appear in the window. By default, this value is set to True. Use PE_UNCHANGED for no change.
hasSearchButton	Specifies whether or not a Search button will appear in the window. By default, this value is set to False. Use PE_UNCHANGED for no change.
hasPrintSetupButton	Specifies whether or not a Print Setup button will appear in the window.
hasRefreshButton	Specifies whether or not a Refresh button will appear in the window.

PEWindowOptions (Other Declares)

VB Type Listing

16-bit

```
Type PEWindowOptions
    StructSize As Integer
    hasGroupTree As Integer
```

```

        canDrillDown As Integer
        hasNavigationControls As Integer
        hasCancelButton As Integer
        hasPrintButton As Integer
        hasExportButton As Integer
        hasZoomControl As Integer
        hasCloseButton As Integer
        hasProgressControls As Integer
        hasSearchButton As Integer
        hasPrintSetupButton As Integer
        hasRefreshButton As Integer
    End Type

```

32-bit

```

Type PEWindowOptions
    StructSize As Integer
    hasGroupTree As Integer
    canDrillDown As Integer
    hasNavigationControls As Integer
    hasCancelButton As Integer
    hasPrintButton As Integer
    hasExportButton As Integer
    hasZoomControl As Integer
    hasCloseButton As Integer
    hasProgressControls As Integer
    hasSearchButton As Integer
    hasPrintSetupButton As Integer
    hasRefreshButton As Integer
End Type

```

Delphi Record Listing

16-bit

```

type
  PEWindowOptions = record
    StructSize: Word;
    hasGroupTree: Smallint;
    canDrillDown: Smallint;
    hasNavigationControls: Smallint;
    hasCancelButton: Smallint;
    hasPrintButton: Smallint;
    hasExportButton: Smallint;
    hasZoomControl: Smallint;
    hasCloseButton: Smallint;

```

```
    hasProgressControls: Smallint;
    hasSearchButton: Smallint;
    hasPrintSetupButton: Smallint;
    hasRefreshButton: Smallint;
end;
```

32-bit

```
type
  PEWindowOptions = record
    StructSize: Word;
    hasGroupTree: Smallint;
    canDrillDown: Smallint;
    hasNavigationControls: Smallint;
    hasCancelButton: Smallint;
    hasPrintButton: Smallint;
    hasExportButton: Smallint;
    hasZoomControl: Smallint;
    hasCloseButton: Smallint;
    hasProgressControls: Smallint;
    hasSearchButton: Smallint;
    hasPrintSetupButton: Smallint;
    hasRefreshButton: Smallint;
  end;
```

PEZoomLevelChangingEventInfo



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

This structure gives information about the PE_ZOOM_LEVEL_CHANGING_EVENT event.

```
struct PEZoomLevelChangingEventInfo
{
  WORD StructSize;
  WORD zoomLevel;
  long windowHandle;
}
```

Members

<i>Member</i>	<i>Description</i>									
StructSize	Specifies the size of the PEZoomLevelChangingEventInfo structure. You must initialize this member to the size of whatever it is, for example, zoomLevelChanging.StructSize = PE_SIZEOF_ZOOM_LEVEL_CHANGNG_EVENT_INFO.									
zoomLevel	The zoom level you wish to set the preview window at. This value can be a value from 25 to 400, indicating a magnification percentage, or it can be one of the following constants:									
<table border="1"><thead><tr><th><i>Constant</i></th><th><i>Value</i></th><th><i>Zoom Level</i></th></tr></thead><tbody><tr><td>PE_ZOOM_SIZE_FIT_ONE_SIDE</td><td>1</td><td>Fit One Side</td></tr><tr><td>PE_ZOOM_SIZE_FIT_BOTH_SIDES</td><td>2</td><td>Fit Both Sides</td></tr></tbody></table>		<i>Constant</i>	<i>Value</i>	<i>Zoom Level</i>	PE_ZOOM_SIZE_FIT_ONE_SIDE	1	Fit One Side	PE_ZOOM_SIZE_FIT_BOTH_SIDES	2	Fit Both Sides
<i>Constant</i>	<i>Value</i>	<i>Zoom Level</i>								
PE_ZOOM_SIZE_FIT_ONE_SIDE	1	Fit One Side								
PE_ZOOM_SIZE_FIT_BOTH_SIDES	2	Fit Both Sides								
<i>Member</i>	<i>Description</i>									
windowHandle	Specifies the frame window handle where the event happens.									

PEZoomLevelChangingEventInfo (Other Declares)

VB Type Listing

16-bit

```
Type PEZoomLevelChangingEventInfo
    StructSize As Integer
    zoomLevel As Integer
    windowHandle As Long
End Type
```

32-bit

```
Type PEZoomLevelChangingEventInfo
    StructSize As Integer
    zoomLevel As Integer
    windowHandle As Long
End Type
```

Delphi Record Listing

16-bit

```
type
  PEZoomLevelChangingEventInfo = record
    StructSize: Word;
    zoomLevel: Integer;
    windowHandle: Longint;
  end;
```

32-bit

```
type
  PEZoomLevelChangingEventInfo = record
    StructSize: Word;
    zoomLevel: Word;
    windowHandle: HWnd;
  end;
```

UXDDiskOptions

Contains file name information that is used by *PEExportOptions*, Page 399, when you want to export to a disk file.

```
struct UXDDiskOptions
{
  WORD structSize;
  char FAR *fileName;
};
```

Members

Member	Description
structSize	Specifies the size of the UXDDiskOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXDDiskOptionsSize.
fileName	Specifies the pointer to the null-terminated string that contains the file name under which you want your disk file saved.

UXDDiskOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXDDiskOptions = record
    structSize: Integer;
    fileName: PChar;
  end;
```

32-bit

```
type
  UXDDiskOptions = record
    structSize: Word;
    fileName: PChar;
  end;
```

UXDMAPIOptions

The **UXDMAPIOptions** structure contains e-mail information that is used by *PEExportOptions*, Page 399, when you want to export to a MAPI e-mail destination.

```
struct UXDMAPIOptions
{
  WORD structSize;

  char FAR *toList;
  char FAR *ccList;
  char FAR *subject;
  char FAR *message;
  WORD nRecipients;
  lpMapiRecipDesc recipients;
}
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXDMAPIOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXDMAPIOptionsSize.

<i>Member</i>	<i>Description</i>
toList	Specifies the pointer to the null-terminated string that contains the “To” list to which you want your e-mail message directed. If you specify “recipients” in this structure, “toList” is ignored.
ccList	Specifies the pointer to the null-terminated string that contains the “CC” list to which you want your e-mail message copied. This string will appear on the message. If you specify “recipients” in this structure, “ccList” is ignored.
subject	Specifies the pointer to the null-terminated string you want to appear as the subject line in the e-mail message.
message	Specifies the pointer to the null-terminated string you want to appear as the body of your e-mail message.
nRecipients	Specifies the number of recipients that are to receive the e-mail message. You must pass the value “0” if you specify “To” and “CC” lists in this structure.
recipients	Specifies the pointer to an array of structures, each of which describes someone to whom the message is being sent or cc'd. This member is included for those applications that are already using Microsoft's MAPI.DLL. If you are not using MAPI.DLL in your application, there is no advantage to using the recipients array over the toList and ccList. You must pass the value “0” if you specify “To” and “CC” lists in this structure. For detailed information about this member, please refer to Microsoft's MAPI documentation.

UXDMAPIOptions (Other Declares)

Delphi Record Listing

16-bit

```

type
  UXDMAPIOptions = record
    structSize: integer;
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
  end;

```

32-bit

```
type
  UXDMAPIOptions = record
    structSize: Word;
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
  end;
```

UXDSMIOptions

The **UXDSMIOptions** structure contains e-mail information that is used by the *PEExportOptions*, [Page 399](#), structure when you want to export to a MAPI e-mail destination.

```
struct UXDSMIOptions
{
  WORD structSize;
  char FAR *toList;
  char FAR *ccList;
  char FAR *subject;
  char FAR *message;
}
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXDMAPIOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXDMAPIOptionsSize.
toList	Specifies the pointer to the null-terminated string that contains the “To” list to which you want your e-mail message directed. If you specify “recipients” in this structure, “toList” is ignored.
ccList	Specifies the pointer to the null-terminated string that contains the “CC” list to which you want your e-mail message copied. This string will appear on the message. If you specify “recipients” in this structure, “ccList” is ignored.
subject	Specifies the pointer to the null-terminated string you want to appear as the subject line in the e-mail message.
message	Specifies the pointer to the null-terminated string you want to appear as the body of your e-mail message.

UXDSMIOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXDSMIOptions = record
    structSize: integer;
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
  end;
```

32-bit

```
type
  UXDSMIOptions = record
    structSize: Word;
    toList: PChar;
    ccList: PChar;
    subject: PChar;
    message: PChar;
  end;
```

UXDPostFolderOptions

Contains information that is used by *PEExportOptions*, [Page 399](#), when you want to export to Microsoft Exchange.

```
struct UXDPostFolderOptions
{
  WORD structSize;
  LPSTR pszProfile;
  LPSTR pszPassword;
  WORD wDestType;
  LPSTR pszFolderPath;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXDPostFolderOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXDPostFolderOptionsSize.
pszProfile	Specifies the Exchange profile.
pszPassword	Specifies the Exchange password.
wDestType	Specifies the type of report to export to. Use one of the following:

<i>Constant</i>	<i>Value</i>	<i>Description</i>
UXDExchFolderType	0	wDestType for Microsoft Exchange folder.
UXDPostDocMessage	1009	wDestType for folder messages.
UXDPostPersonalReport	1010	wDestType for personal report.
UXDPostFolderReport	1011	wDestType for folder report.

<i>Member</i>	<i>Description</i>
pszFolderPath	Specifies the Exchange path where you want the program to place the exported file.

UXDPostFolderOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXDPostFolderOptions = record
    structSize: Integer;
    pszProfile: PChar;
    pszPassword: PChar;
    wDestType: Integer;
    pszFolderPath: PChar;
  end;
```

(*pszFolderPath has to be in the following format: <Message Store Name>@<Folder Name>@<Folder Name>*)

32-bit

```
type
  UXDPostFolderOptions = record
    structSize:Word;
    pszProfile:PChar;
    pszPassword:PChar;
    wDestType: Word;
    pszFolderPath:PChar;
  end;
```

(*pszFolderPath has to be in the following format: <Message Store Name>@<Folder Name>@<Folder Name>*)

UXDVIMOptions

Contains e-mail message information that is used by *PEExportOptions*, *Page 399*, when you want to export a VIM e-mail destination.

```
struct UXDVIMOptions
{
  WORD structSize;
  char FAR *toList;
  char FAR *ccList;
  char FAR *bccList;
  char FAR *subject;
  char FAR *message;
};
```

NOTE: VIM is not supported in the 32-bit version of Seagate Crystal Reports.

Members

Member	Description
structSize	Specifies the size of the UXDVIMOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXDVIMOptionsSize.
toList	Specifies the pointer to the null-terminated string that contains the “To” list to which you want your e-mail message directed.
ccList	Specifies the pointer to the null-terminated string that contains the “CC” list to which you want your e-mail message copied. This string will appear on the message.

<i>Member</i>	<i>Description</i>
bccList	Specifies the pointer to the null-terminated string that contains the “Blind CC” list to which you want your e-mail message copied. This string will not appear on the message.
subject	Specifies the pointer to the null-terminated string you want to appear as the subject line in the e-mail message.
message	Specifies the pointer to the null-terminated string you want to appear as the body of your e-mail message.

UXDVIMOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXdVIMOptions = record
    structSize: integer;
    toList: PChar;
    ccList: PChar;
    bccList: PChar;
    subject: PChar;
    message: PChar;
  end;
```

UXFCharSeparatedOptions

Contains number and date information used by the *PEExportOptions*, Page 399, structure when you want to export in a Character Separated format and hard code number and/or date options.

```
struct UXFCharSeparatedOptions
{
  WORD structSize;
  BOOL useReportNumberFormat;
  BOOL useReportDateFormat;
  char stringDelimiter;
  char FAR *fieldDelimiter;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFCharSeparatedOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFCharSeparatedOptionsSize.
useReportNumberFormat	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass TRUE if you want the program to use the same format used in the report, FALSE if you want the number saved in a format that has been optimized for the file format you have selected.
useReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass TRUE if you want the program to use the same format as used in the report, FALSE if you want dates saved in a format that has been optimized for the file format you have selected.
stringDelimiter	Specifies the character you want to use to enclose alphanumeric field data in the character separated values format. You can use whatever character you wish, and it must be enclosed in quotes.
fieldDelimiter	Specifies a pointer to the string you want to use to separate the fields in the character separated values format. Your string may be up to 16 characters long and must be enclosed in quotes.

UXFCharSeparatedOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFCharSeparatedOptions = record
    structSize: integer;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
    stringDelimiter: Char;
    fieldDelimiter: PChar;
  end;
```

32-bit

```
type
  UXFCharSeparatedOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
    stringDelimiter: Char;
    fieldDelimiter: PChar;
  end;
```

UXFCommaTabSeparatedOptions

Contains number and date information used by *PEExportOptions*, *Page 399*, when you want to export in a Comma-separated or Tab-separated format and hard code number and/or date options.

```
struct UXFCommaTabSeparatedOptions
{
  WORD structSize;
  BOOL useReportNumberFormat;
  BOOL useReportDateFormat;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFCommaTabSeparatedOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFCommaTabSeparatedOptionsSize.
useReportNumberFormat	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass TRUE if you want the program to use the same format used in the report, FALSE if you want the number saved in a format that has been optimized for the file format you have selected.
useReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass TRUE if you want the program to use the same format as used in the report, FALSE if you want dates saved in a format that has been optimized for the file format you have selected.

UXFCommaTabSeparatedOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFCommaTabSeparatedOptions = record
    structSize: integer;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

32-bit

```
type
  UXFCommaTabSeparatedOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

UXFDIFOptions

Contains number and date information used by *PEExportOptions*, *Page 399*, when you want to export in a DIF format (Data Interchange Format) and hard code number and/or date options.

```
struct UXDIFOptions
{
  WORD structSize;
  BOOL useReportNumberFormat;
  BOOL useReportDateFormat;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFDIFOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFDIFOptionsSize.

<i>Member</i>	<i>Description</i>
useReportNumberFormat	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass TRUE if you want the program to use the same format used in the report, FALSE if you want the number saved in a format that has been optimized for the file format you have selected.
useReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass TRUE if you want the program to use the same format as used in the report, FALSE if you want dates saved in a format that has been optimized for the file format you have selected.

UXFDIFOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFDCommaTabSeparatedOptions = record
    structSize: integer;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

32-bit

```
type
  UXFDIFOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

UXFHTML3Options

Used by *PEExportOptions*, *Page 399*, when you are exporting to HTML format.

```
struct UXFHTML3Options
{
    WORD structSize;
    char FAR *fileName;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFHTML3Options structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFHTML3OptionsSize.
fileName	Specifies a null terminated file name. For example, "C\pub\docs\boxoffic\default.htm". Any exported GIF files will be located in the same directory as this file.

UXFHTML3Options (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFHTML3Options = record
    structSize: Integer;
    fileName: PChar;
  end;
```

32-bit

```
type
  UXFHTML3Options = record
    structSize: Word;
    fileName: PChar;
  end;
```

UXFODBCOptions

Contains information used by *PEExportOptions*, *Page 399*, whenever you export in ODBC format.

```
struct UXFODBCOptions
{
    WORD      structSize;
    char FAR *dataSourceName;
    char FAR *dataSourceUserID;
    char FAR *dataSourcePassword;
    char FAR *exportTableName;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFODBCOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFODBCOptionsSize.
dataSourceName	Specifies the name of the data source that you want to export to.
dataSourceUserID	Specifies the User ID that you need to connect to the data source.
dataSourcePassword	Specifies the Password that you need to connect to the data source.
exportTableName	Specifies the name of the table you want to export to in the data source.

UXFODBCOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
    UXFODBCOptions = record
        structSize: Integer;
        dataSourceName: PChar;
        dataSourceUserID: PChar;
        dataSourcePassword: PChar;
        exportTableName: PChar;
    end;
```

32-bit

```
type
  UXFODBCOptions = record
    structSize: Word;
    dataSourceName: PChar;
    dataSourceUserID: PChar;
    dataSourcePassword: PChar;
    exportTableName: PChar;
  end;
```

UXFPaginatedTextOptions

Contains information used by *PEExportOptions*, *Page 399*, whenever you export in paginated text format.

```
struct UXFPaginatedTextOptions
{
  WORD structSize;
  WORD nLinesPerPage;
};
```

Members

<i>Member</i>	<i>Description</i>
structSize	Specifies the size of the UXFPaginatedTextOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFPaginatedTextOptionsSize.
nLinesPerPage	Indicates the number of lines to be printed before the page break. The default is 60 lines. When the paginated text format is used with <i>PEGetExportOptions</i> , <i>Page 155</i> , the program displays the Lines Per Page dialog box to give the user the opportunity to specify a different number if he or she wishes.

UXFPaginatedTextOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFPaginatedTextOptions = record
    structSize: Integer;
    nLinesPerPage: Integer;
  end;
```

32-bit

```
type
  UXFPaginatedTextOptions = Record
    structSize:Word;
    nLinesPerPage:Word;
  end;
```

UXFRecordStyleOptions

Contains number and date information used by *PEExportOptions*, *Page 399*, when you want to export in a Record style (columns of values) format and hard code number and/or date options.

```
struct UXFRecordStyleOptions
{
  WORD structSize;
  BOOL useReportNumberFormat;
  BOOL useReportDateFormat;
};
```

Members

Member	Description
structSize	Specifies the size of the UXFRecordStyleOptions structure. You must initialize this member to be the size of whatever it is, for example, options.structSize = UXFRecordStyleOptionsSize.

<i>Member</i>	<i>Description</i>
useReportNumberFormat	Indicates whether or not the program should save numbers in the same format (decimal places, negatives, etc.) that you have used in the report. Pass TRUE if you want the program to use the same format used in the report, FALSE if you want the number saved in a format that has been optimized for the file format you have selected.
useReportDateFormat	Indicates whether or not the program should save dates in the same format (MDY, DMY, etc.) that you used in the report. Pass TRUE if you want the program to use the same format as used in the report, FALSE if you want dates saved in a format that has been optimized for the file format you have selected.

UXFRecordStyleOptions (Other Declares)

Delphi Record Listing

16-bit

```
type
  UXFRecordStyleOptions = record
    structSize: integer;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

32-bit

```
type
  UXFRecordStyleOptions = record
    structSize: Word;
    useReportNumberFormat: Bool;
    useReportDateFormat: Bool;
  end;
```

8

Crystal ActiveX Control Reference

What you will find in this chapter...

Section Codes (32-bit), Page 494

PROPERTIES, Page 496

METHODS, Page 594

Section Codes (32-bit)

NOTE: If you are running a 16-bit system, see Section Codes (16-bit), Page 495.

When a section code is required, use the following syntax to supply the values the program needs in order to generate the code:

```
section type.group.section number
```

Section type

Section	Section type
Report Header Section	REPORTHDR
Page Header Section	PAGEHDR
Group Header Section	GROUPHDR
Detail Section	DETAIL
Group Footer Section	GROUPFTR
Report Footer Section	REPORTFTR
Page Footer Section	PAGEFTR

If the report contains one or more groups and you are specifying a group section, use an integer to specify the group of interest (the first group is Group 0, the second is Group 1, and so forth).

NOTE: If you are specifying the section code for anything other than a group section, use 0 for this parameter.

Section

If an area contains more than one section, use an integer to specify the section of interest (the first section in an area is Section 0, the second is Section 1, and so forth).

Example section codes

Specifying the only (or the first) section in a non-group area

```
REPORTHDR.0.0
```

«The first Report Header section.»

Specifying one of several sections in a non-group area

If a report has multiple Detail sections and you want to specify the fourth Detail section, you can specify that section as:

DETAIL.0.3

«The fourth Detail section.»

Specifying the only (or the first) section in a group area

GROUPHDR.0.0

«The first section in the first Group Header area.»

Specifying one of several sections in a group area

GROUPFTR.2.1

«The second section in the third Group Footer area.»

Section Codes (16-bit)

Select from the following section codes whenever you are asked to provide a section code value as a parameter for one of the Crystal ActiveX Control properties:

<i>To specify this section</i>	<i>Use this code</i>
All sections	ALL
Title section	TITLE
Page Header section	HEADER
Group Header 1 section	GH1
Group Header 2 section	GH2
Group Header 3 section	GH3
Group Header 4 section	GH4
Group Header 5 section	GH5
Group Header 6 section	GH6
Group Header 7 section	GH7
Group Header 8 section	GH8
Group Header 9 section	GH9
Details section	DETAIL
Group Footer 1 section	GF1
Group Footer 2 section	GF2
Group Footer 3 section	GF3
Group Footer 4 section	GF4
Group Footer 5 section	GF5
Group Footer 6 section	GF6

<i>To specify this section</i>	<i>Use this code</i>
Group Footer 7 section	GF7
Group Footer 8 section	GF8
Group Footer 9 section	GF9
Grand Total section	GRNDTTL
Page Footer section	FOOTER

PROPERTIES

Action

Description

Triggers the printing of the report.

Usage

```
[form.]Report.Action = 1
```

For example:

```
CrystalReport1.Action = 1
```

«Prints the specified report.»

Remarks

Set the Action property to 1 in your procedure code (CrystalReport1.Action = 1) to print the report in response to a user event.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEStartPrintJob, Page 348

BoundReportFooter

Description

Indicates whether or not a footer is printed at the bottom of each page with a page number when printing a bound report.

Usage

```
[form.]Report.BoundReportFooter [= {True|False}]
```

For example:

```
CrystalReport1.BoundReportFooter = True
```

«Displays a footer at the bottom of each page of the bound report with a page number.»

Remarks

- This property is ignored if the *ReportSource*, *Page 559*, is set to 0.
- The default value for this property is False.
- This property/method is available for subreports.

Data Type

Boolean

Availability

Design Time; Runtime

BoundReportHeading

Description

Specifies a report title to be displayed at the top of the first page of a bound report.

Usage

```
[form.]Report.BoundReportHeading [= Title$]
```

For example:

```
CrystalReport1.BoundReportHeading = "Box Office Report"
```

«Specifies that the title “Box Office Report” be printed at the top of the first page of the report.»

Remarks

- This property is ignored if the *ReportSource*, *Page 559*, is set to 0.
- If this property is left blank, no report title will be printed.

Data Type

String

Availability

Design Time; Runtime

Connect

Description

Logs on to an SQL server or an ODBC data source.

Usage

```
[form.]Report.Connect [=  
DataSourceName;UserID;Password;DatabaseQualifier$]
```

For example:

```
CrystalReport1.Connect = "DSN = Accounting;UID = 734;PWD = bigboard;DSQ  
= Administration"
```

«Connects to the “Administration” database on the “Accounting” server using the user ID #734 and the password “bigboard”.»

```
CrystalReport1.Connect = "dsn=; uid=; pwd=bigboard; dsq="
```

«Connects to a password-protected Paradox database. All that is being passed is the password “bigboard”.»

Remarks

- Use the Connect property when the report connects to only a single ODBC data source or SQL server, and only a single set of log on information is required. If the report connects to multiple data sources that require different log on information, use *LogOnInfo*, Page 526.
- Enter the parameters necessary to log on to the SQL server that you need to be activated for your report. Parameters should be in the following format:

DSN = name;UID = userID;PWD = password;DSQ = database qualifier

 - *name* is the server name or ODBC data source name,
 - *userID* is the name you have been assigned for logging on to the SQL server,
 - *password* is the password you have been assigned for logging on to the SQL server, and
 - *database qualifier* is the database name if your server uses the database concept.
- The database qualifier parameter, DSQ, is required only when it is applicable to the ODBC / SQL driver you are using. If your DBMS does not use the database concept, you do not need to specify the DSQ parameter.
- Before you can use this property for an ODBC/SQL database, you must install the ODBC / SQL driver for whatever SQL database you are planning to use, and put the Database/BIN location in your path.
- If you are connecting to an SQL or other password protected database directly, without going through ODBC, use the name of the SQL server for the DSN parameter.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESetNthTableLogOnInfo, Page 320

CopiesToPrinter

Description

Specifies the number of copies to be printed if you are printing to a printer, if the value you assign *Destination*, *Page 502*, a value of 1-Printer.

Usage

```
[form.]Report.CopiesToPrinter[ = NumCopies%]
```

For example:

```
CrystalReport1.CopiesToPrinter = 3
```

«Prints three copies of the specified report.»

Remarks

- The number you enter must not be a zero or a negative value.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToPrinter, Page 254

DataFiles

Description

Specifies the location of the database files or tables used in the report.

Usage

```
[form.]Report.DataFiles(ArrayIndex) [= Location$]
```

- Enter the file name and path of each database file or table in your report for which you want to change the location.
- Use a separate line of code for each file for which you want to change the location.
- The order of files in the array must conform to the order of files in the report. (You can use the Database | Set Location command to determine the order of files in the report.)
- The first file in the report is array index (0), the second file is (1), etc.

For example, to change the location of the first and third files in a report (first.dbf and third.dbf) to the c:\new directory, use the following syntax:

```
CrystalReport1.DataFiles(0) = "c:\new\first.dbf"  
CrystalReport1.DataFiles(2) = "c:\new\third.dbf"
```

Remarks

- DataFiles is an array property that is available at runtime only.
- Use this property if you want to run the report with files in different locations than specified in the report.
- When using this property, you do not have to change the locations of all files in the report. Just make certain that the array index for each file you do change matches the position of that file in the report.
- This property is cleared once the print job is printed. If you print a second time, the program reverts to the locations as originally specified in the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetNthTableLocation, Page 319

DataSource

Description

Specifies the Data Control that the bound report will retrieve data from.

Remarks

- This property is ignored if the *ReportSource*, *Page 559*, is set to 0.
- The Data Control must exist on the form before this property can be set.
- This property is only available at design time.
- This property/method is available for subreports.

Data Type

Data Control object

Availability

Design Time

Destination

Description

Specifies the destination to which your report is to be printed (Window, Printer, File or Mail).

Usage

```
[form.]Report.Destination[ = Destination%]
```

For example:

```
CrystalReport1.Destination = 0
```

«Sends the specified report to print to a window.»

Remarks

Select one of the following print destinations:

- 0 = Window (Sends the report to a preview window.)

- 1 = Printer (Sends the report to a printer.)
- 2 = File (Prints the report to a disk file for printing at a later time or for importing into other applications. If you select this property, you will also have to set the *PrintFileName*, *Page 543*, and the *PrintFileType*, *Page 547*.)
- 3 = E-mail via MAPI (Sends the report to another person on your network via MAPI e-mail (Microsoft Mail). The report is attached to the e-mail letter in the format specified by the *PrintFileType* property.)
- 4 = E-mail via VIM (Sends the report to another person on your network via VIM e-mail (cc:Mail). The report is attached to the e-mail letter in the format specified by the *PrintFileType* property.)
- 5 = To Notes (Sends the report to a Lotus Notes destination.)
- 6 = To Exchange Folder (Sends the report to a Microsoft Exchange folder).

This property/method is available for subreports.

Data Type

Integer (Enumerated)

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToPrinter, *Page 254*

PEOutputToWindow, *Page 257*

PEExportTo, *Page 146*

DetailCopies

Description

Specifies the number of copies of each record in the Details section that the program is to print.

Usage

```
[form.]Report.DetailCopies [= NumCopies%]
```

For example:

```
CrystalReport1.DetailCopies = 3
```

«Specifies that three (3) copies of each record in the details section are to be printed.»

Remarks

- If DetailCopies is set to a value less than or equal to zero, the value is ignored and 1 copy of the Detail section of the report is printed.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetNDetailCopies, Page 309

DialogParentHandle

Description

Specifies the handle of the parent window. The program uses this handle to determine the window within which it centers any dialog boxes it displays (progress dialog boxes, parameter field prompt dialog boxes, and so forth).

Usage

```
[form.]Report.DialogParentHandle = [HWND]
```

For example:

```
CrystalReport1.DialogParentHandle = ParentHwnd
```

«Specifies the handle of the parent for all dialog boxes that the control will display.»

Remarks

- Does not affect the placement of the preview window. Preview window placement is determined by the WindowLeft, WindowTop, WindowHeight, and WindowWidth properties.
- This property/method is available for subreports.

Data Type

Long Integer

Availability

Runtime only

Related Report Engine Functions

PESetDialogParentWindow, Page 274

DiscardSavedData

Description

If data is saved with the specified report, setting this property to 1 (True) discards the data.

Usage

```
[form.]Report.DiscardSavedData [= TrueFalse%]
```

For example:

```
CrystalReport1.DiscardSavedData = 1
```

«Discards the data saved with the specified report.»

Remarks

For TrueFalse% use one of the following values:

- False = 0
- True = 1

This property/method is available for subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEDiscardSavedData, Page 140

EMailCCList

Description

Specifies the “CC” list to which you want your e-mail message sent.

Usage

```
[form.]Report.EMailCCList [= CCList$]
```

For example:

```
CrystalReport1.EMailCCList = "John Brown; Jane Doe"  
«Sends a CC of the e-mail message to both John Brown and Jane Doe.»
```

Remarks

- Applies to both VIM and MAPI.
- Multiple names must be separated by a semicolon.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

PEGetExportOptions, Page 155

EMailMessage

Description

Specifies the string you want to appear as the body of your e-mail message.

Usage

```
[form.]Report.EMailMessage [=Message$]
```

For example:

```
CrystalReport1.EMailMessage = "The meeting is at 4:00"
```

«Sets "The meeting is at 4:00" as the body of the e-mail message.»

Remarks

- Applies to both MAPI and VIM.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

PEGetExportOptions, Page 155

EMailSubject

Description

Specifies the subject line in your e-mail message.

Usage

```
[form.]Report.EMailSubject [= Subject$]
```

For example:

```
CrystalReport1.EMailSubject = "Staff meeting"
```

«Sets "Staff meeting" as the subject line in an e-mail message.»

Remarks

- Applies to both MAPI and VIM.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

EMailToList

Description

Specifies the "To" list to which you want your e-mail message directed.

Usage

```
[form.]Report.EMailToList [=ToList$]
```

For example:

```
CrystalReport1.EMailToList = "Jane Doe"  
«Makes Jane Doe the only name in the To list.»
```

Remarks

- Applies to both MAPI and VIM.
- Multiple names must be separated by a semicolon.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

EMailVIMBCCList

Description

Specifies the “Blind CC” list to which you want your e-mail message copied.

Usage

```
[form.]Report.EMailVIMBCCList [=BCCList$]
```

For example:

```
CrystalReport1.EMailVIMBCCList = "John Jacobs; Jane Doe"  
«Makes John Jacobs and Jane Doe the names for the BCC list.»
```

Remarks

- Applies to VIM only, not MAPI.
- Multiple names must be separated by a semicolon.

- This property is not available in the 32-bit ActiveX control.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

ExchangeFolder

Description

Specifies the Exchange path to export a file, when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangeFolder [= ExchangeFolderPath$]
```

For example:

```
CrystalReport1.ExchangeFolder = "c:\Microsoft\Exchange\newrpt.rpt"
```

```
«Send the report to file "newrpt.rpt" in the subdirectory \Microsoft\Exchange.»
```

NOTE: ExchangeFolder is case-sensitive. If you enter a value in the wrong case you will receive an error message.

Data Type

String

Availability

Runtime

Remarks

This property/method is available for subreports.

Related Report Engine Functions

PEExportTo, Page 146

ExchangePassword

Description

Specifies the Exchange password when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangePassword [= Password$]
```

For example:

```
CrystalReport1.ExchangePassword = "pickle"
```

«The Exchange password is “pickle”.»

Data Type

String

Availability

Runtime

Remarks

This property/method is available for subreports.

Related Report Engine Functions

PEExportTo, Page 146

ExchangeProfile

Description

Specifies the Exchange Profile when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangeProfile [= Profile$]
```

For example:

```
CrystalReport1.ExchangeProfile = "James Andrews"
```

«Specifies "James Andrews" as the Exchange Profile.»

Remarks

- Usually your profile is your name.
- This property/method is available for subreports.

Data Type

String

Availability

Runtime

Related Report Engine Functions

PEExportTo, Page 146

Formulas

Description

Specifies a new string for an existing formula.

Usage

```
[form.]Report.Formulas(ArrayIndex) [= "FormulaName= FormulaText"]
```

Enter the formula name and the string that you want to replace the existing string for each formula that you want to change in your report.

For example, to change a formula @COMMISSION to {file.SALES}*1, and a second formula @TOTAL to {file.SALES} + {file.COMMISSION}, enter the following:

```
CrystalReport1.Formulas(0) = "COMMISSION= {file.SALES} *.1"
```

```
CrystalReport1.Formulas(1) = "TOTAL= {file.SALES} + {file.COMMISSION}"
```

Remarks

- Formulas is an array property that is available at runtime only.
- Use a separate line of code for each formula you want to change.
- Change only those formulas that you want to change.
- The first formula you change must be assigned array index (0), the second must be assigned array index (1), etc.
- The new formula string must conform to Seagate Crystal Reports syntax requirements.
- This property is cleared once the print job is printed. If you print a second time, the program reverts to the formulas as originally specified in the report.

NOTE: Spaces are significant in formula names. For this reason, the equals sign must follow the formula name with no intervening spaces.

NOTE: The @ sign is not used when designating a formula name in this property.

NOTE: You can't use this property to create new formulas. You can only use it to change existing formulas.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetFormula, Page 287

GraphData

Description

Sets the data used for a specified graph.

Usage

```
[form.]Report.GraphData(ArrayIndex%)  
[= sectionCode; graphNum; row; col;  
field;direction$]
```

For example:

```
CrystalReport1.GraphData(0)= "GROUPHDR.0.0; 1; GROUP2; GROUP1; 0;  
COLANDROW"
```

«The value in Group 1 is used for the rows of the graph, the value in Group 2 is used for the columns of the graph, the first summarized field added to the report is used to set the value of the risers of the graph, and values in both columns and rows are used to create the graph.»

NOTE: *This example uses 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.*

Remarks

With GraphData, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action=1” call.

The array index value for GraphData simply specifies the sequence number for the change. Thus:

```
CrystalReport1.GraphData(0) = "GROUPHDR.0.0; 3; Group1; Group2; 666;  
COLANDROW"
```

when making changes to one graph only, but

```
CrystalReport1.GraphData(0) = "HEADER; 3; Group1; Group2; 666;  
COLANDROW" CrystalReport1.GraphData(1) = "GROUPHDR.0.0; 3; Group1;  
Group2; 666; COLANDROW"
```

when making changes to more than one graph.

NOTE: *These examples use 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.*

Use the following table as a guide in supplying the required values for this property:

Parameter	Description	Expected value
sectionCode	Specifies the section in which you want to modify a graph.	Please refer to <i>Section Codes (32-bit), Page 494</i> .
graphNum	The number of the graph within the section you want to modify.	Graphs in a section are numbered, starting with zero, left to right first, then top to bottom.
row	The Group number in the report used to create rows in the graph.	GROUP1, GROUP2, GROUP3,..., GROUP9
col	The Group number in the report used to create columns in the graph.	GROUP1, GROUP2, GROUP3,..., GROUP9

<i>Parameter</i>	<i>Description</i>	<i>Expected value</i>
field	The summarization field containing values to be used as the value of each riser in the graph.	The first summary field added to a report is numbered 0, the second is numbered 1, etc.
direction	Whether the values in the rows, the columns, or both are used to create the graph.	ROWS, COLS, ROWANDCOL, or COLANDROW

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetGraphData, Page 289

GraphOptions

Description

Sets a number of options for the specified graph.

Usage

```
[form.]Report.GraphOptions(ArrayIndex%) [= sectionCode; graphNum;
fontFace; barDirection; labelRisers; gridLines; legend; max; min$]
```

For example:

```
CrystalReport1.GraphOptions(0) = "FOOTER;0;Arial;H;T;F;X;max;min"
```

«Sets the font to Arial, sets horizontal bars, shows a data value on every riser (labelRisers = T), and toggles the grid lines off in the first Graph in the Page Footer section.»

Remarks

With GraphOptions, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action=1” call. The array index value for GraphOptions simply specifies the sequence number for the change. Thus:

```
CrystalReport1.GraphOptions(0) = "GROUHDR.0.0; 1; Arial; (H; T; F;  
legend; max; min"
```

when making changes to one graph only, but

```
CrystalReport1.GraphOptions(0) = "TITLE; 1 Arial; H; T; F; X; 100; 0"
```

```
CrystalReport1.GraphOptions(1) = "TITLE; 1 Arial; H; T; F; X; 100; 0"
```

when making changes to more than one graph.

NOTE: These examples use 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.

Use the following chart as a guide in entering the required property values:

Parameter	Description	Values expected
sectionCode	Specifies the section in which you want to modify a graph.	Please refer to <i>Section Codes (32-bit)</i> , Page 494s.
graphNum	Specifies which graph in the section you want to modify.	Graphs in a section are numbered, starting with zero, left to right first, then top to bottom.
fontFace	Specifies the font face you want to use for the entire graph.	Actual name of font (i.e., Arial).
barDirection	In a bar graph, specifies the direction in which you want the graph bars to appear.	H = horizontal, V = vertical, X = as is
labelRisers	Specifies whether or not you want to show the data value on every riser.	T= true, F = False, X = as is
gridLines	Specifies whether or not you want to show grid lines.	T= true, F = False, X = as is
legend	Specifies whether or not you want to show a legend.	T= true, F = False, X = as is
max	Specifies the maximum value you want included in your graph.	Enter a number.

<i>Parameter</i>	<i>Description</i>	<i>Values expected</i>
min	Specifies the minimum value you want included in your graph.	Enter a number.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetGraphOptions, Page 291

GraphText

Description

Sets the various text components for the specified graph.

Usage

```
[form.]Report.GraphText(ArrayIndex%) [= sectionCode; graphNum; title;
subTitle; footnote; series; group; x; y; z$]
```

For example:

```
CrystalReport1.GraphText(0) = "HEADER; 0;;;;;new x label; new y
label;new z label"
```

«Resets the x, y, and z labels for the first graph in the Page Header section.»

Remarks

- Select your section code from the section code table (see *Section Codes (32-bit), Page 494*).
- With GraphText, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action=1” call. The array index value for GraphText simply specifies the sequence number for the change. Thus:

```
CrystalReport1.GraphText(0) = " GROUPHDR.0.0; 1; title string;  
subtitle string; footnote string; series string; group string;  
x string;y string; z string"
```

when making changes to one graph only, but

```
CrystalReport1.GraphText(0) = "TITLE; 1; title string; subtitle  
string; footnote string; series string; group string; x string;  
y string; z string"
```

```
CrystalReport1.GraphText(1) = "TITLE; 1; title string; subtitle  
string; footnote string; series string; group string; x string;  
y string; z string"
```

when making changes to more than one graph.

NOTE: These examples use 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.

- *title, subTitle, footnote, series, group, x, y, and z* are the strings you want to label the appropriate parts of the graph.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetGraphText, Page 292

GraphType

Description

Sets the kind of graph used in the selected section in the specified report.

Usage

```
[form.]Report.GraphType(ArrayIndex%) [=sectionCode;graphNum;graphType$]
```

For example:

```
CrystalReport1.GraphType(0) = "GROUPHDR.0.0; 0; PIE"
```

«Specifies a Pie graph as the first graph (graphNum =0) in the Group Header 1 section.»

NOTE: This example uses 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.

Remarks

- With GraphType, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action=1” call.

The array index value for GraphType specifies the sequence number for the change. Thus:

```
CrystalReport1.GraphType(0) = "GROUPHDR.0.0; 0; PIE"
```

when making changes to one graph only, but

```
CrystalReport1.GraphType(0) = "HEADER; 0; PIE"
```

```
CrystalReport1.GraphType(1) = "GROUPHDR.0.0; 0; PIE"
```

when making changes to more than one graph.

NOTE: These examples use 32-bit section codes. If you are running a 16-bit system, see Section Codes (16-bit), Page 495.

- Select sectionCode from the section code table, see *Section Codes (32-bit), Page 494*.
- Graph numbers are 0 origin; the first graph in a section is number 0, the second is number 1, etc.
- Multiple graphs in a section are numbered left to right first, then top to bottom.
- Select from the following graph types for the GraphType value for this property:

<i>For this type of graph</i>	<i>Use this code for graphType</i>
Side-by-side	SIDEBYSIDE
3-D side-by-side	3DSIDE
Stacked bar	STACKEDBAR
3-D stacked bar	3DSTACKED
Percent bar	PERCENTBAR
3-D percent bar	3DPERCENT
Line	LINE
Area	AREA
3-D bars	3DBARS
Pie	PIE

<i>For this type of graph</i>	<i>Use this code for graphType</i>
Multiple pie	MULTIPLEPIE
Weighted pie	WEIGHTEDPIE

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

Related Report Engine Functions

PESetGraphType, Page 294

GroupCondition

Description

Specifies what kind of change in the Group Condition Field will trigger the creation of a group.

Usage

```
[form.]Report.GroupCondition(SequentialIndex%)
[= group; field; condition; sortDirection$]
```

For example:

```
CrystalReport1.GroupCondition(0) = "GROUP1;{order details.ORDER
ID};ANYCHANGE;A"
```

«Specifies that any change in the *ordernum* field in Group1 will trigger a new grouping.»

Remarks

Refer to the following tables for parameter values for this property:

Parameter	Description	Values expected
group	The group in which you want to set the group condition.	The outermost group on the report is GROUP1, the next group is GROUP2, etc.
field	The name of the field that triggers a grouping whenever its value changes.	Enter the name in the following format: {table.FIELDNAME}
condition	Enter the condition that triggers the grouping.	See the tables below.
sortDirection	The direction in which groups are to be sorted.	A = Ascending, D = Descending.

Condition (Date Fields)	Condition Code
Daily	DAILY
Weekly	WEEKLY
Bi-weekly	BIWEEKLY
Semi-monthly	SEMIMONTTHLY
Monthly	MONTHLY
Quarterly	QUARTERLY
Semi-annually	SEMIANNUALLY
Annually	ANNUALLY

Condition (Boolean Fields)	Condition Code
To Yes	TOYES
To No	TONO
Every Yes	EVERYYES
Every No	EVERYNO
Next Is Yes	NEXTISYES
Next Is No	NEXTISNO

Condition for all other data types	
Any Change	ANYCHANGE

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime only

Related Report Engine Functions

PESetGroupCondition, Page 296

GroupSelectionFormula

Description

Specifies the groups to be used when printing the report.

Usage

```
[form.]Report.GroupSelectionFormula  
[= "GroupSelectionFormula"]
```

Enter the group selection formula just as you would enter it in the Formula Editor. For example, to limit your report to those groups with a subtotal on the {order details.ORDER AMOUNT} field less than \$10,000 (with subtotals triggered by changes in the {customer.CUSTOMER ID} field), you would enter the following as a group selection formula:

```
Sum ({order details.ORDER AMOUNT}, {customer.CUSTOMER ID}) < $10000
```

Remarks

If your group selection formula includes internal quotes, change all of the internal double quotes to single quotes and then surround the entire selection formula in double quotes.

NOTE: If you have created a group selection formula in your report at Design Time, any group selection formula you enter here will be appended to that group selection formula, connected by an "and". Thus, your records will be selected based on a combination of the two formulas.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESetGroupSelectionFormula, Page 303

GroupSortFields

Description

Specifies the group field(s) that are to be used to sort your data when the report is printed.

Usage

```
[form.]Report.GroupSortFields(ArrayIndex)  
[= "+|-GroupField"]
```

Enter the group field(s) on which you want your report to be sorted.

For example, assume that you have broken your data into state groups and had Seagate Crystal Reports count the number of customers in each group. In order to print the group with the highest count first, then the group with the next highest count, etc. (descending order), enter a string similar to the following:

```
CrystalReport1.GroupSortFields(0) = "-Count  
({customer.CUSTOMER_ID},{customer.REGION})"
```

Remarks

- GroupSortFields is an array property available at runtime only.
- Use a separate line of code to specify each group sort field.
- Enter group sort fields in the order that you want them to sort your report. For example, if you want your report to be sorted first on group sort field A and then on group sort field B, specify group sort field A in your first line of code and group sort field B in your second line of code.
- The first group sort field you specify must be assigned array index 0, the second group sort field must be assigned array index 1, etc.
- The index values you assign must be continuous; no gaps are allowed (0, 1, 2 = OK, 0, 1, 3 = wrong).
- Array index values must be subscripted in the code immediately after the property name (i.e., CrystalReport1.GroupSortFields(0) =).
- If you have specified sort fields for your report at Design Time, any sort fields you enter here will replace the sort fields in your report.
- If you do not use this property, the program will use the sorting instructions that you specified in the report.

- If you want to clear the group sort fields in your report, use an empty string (i.e., CrystalReport1.GroupSortFields(0) = "").
- This property is cleared once the print job is printed. If you print a second time, the program reverts to the group sort fields as originally specified in the report.

NOTE: The group sort field entry must follow the sort direction sign (+ or -) with no intervening space.

NOTE: To find the correct syntax for any group in your report using Seagate Crystal Reports for Visual Basic:

- choose the *Formula Field* command from the *Insert* menu,
- enter any formula name in the *Insert Formula* dialog box when it appears,
- click the scroll button on the *Fields* list in the *Formula Editor* when it appears, and
- double-click the group field of interest.

Seagate Crystal Reports enters the group field name in the Formula text box. Use the name and syntax from that text box when constructing your group sort field string.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetNthGroupSortField, Page 311

LastErrorMessage

Description

Returns the error code for the last runtime error.

Usage

[form.]Report.LastErrorMessage

For example:

```
'If error occurs, go to Error Handler  
On Error GoTo ErrorHandler  
  
ErrorHandler:  
    MsgBox CrystalReport1.LastErrorNumber  
  
«If an error occurs, this code calls up a message box that displays the error number.»
```

Remarks

- LastErrorNumber is a runtime-only property.
- This property/method is available for subreports.

NOTE: LastErrorNumber must come after the Action call in order to display relevant values. After you have printed your report, you can refer to this property to get an error number (if any). If there was no error in printing, LastErrorNumber = 0.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEGetErrorCode, Page 152

LastErrorMessage

Description

Returns the error string for the last runtime error.

Usage

```
[form.]Report.LastErrorMessage
```

For example:

```
'If error occurs, go to Error Handler  
On Error GoTo ErrorHandler
```

```
ErrorHandler:  
    MsgBox CrystalReport1.LastErrorString
```

«If an error occurs, this code calls up a message box that displays the error string.»

Remarks

- LastErrorString is a runtime-only property.
- This property/method is available for subreports.

NOTE: LastErrorString must come after the Action call in order to display relevant values. After you have printed your report, you can refer to this property to get an error string (if any). If there was no error in printing, LastErrorNumber = 0.

Data Type

String

Availability

Runtime

Related Report Engine Functions

PEGetErrorCode, Page 152

PEGetErrorText, Page 153

LogOnInfo

Description

LogOnInfo logs on to one or more SQL servers or a password-protected databases.

Usage

```
[form.]Report.LogOnInfo(ArrayIndex)[ =  
Name;UserID;Password;DatabaseQualifier$]
```

For example:

```
CrystalReport1.LogOnInfo[0] = "DSN = Accounting;UID = 734;PWD =  
bigboard;DSQ = Administration"
```

«Connects to the “Administration” database on the “Accounting” server using the user ID #734 and the password “bigboard”.»

```
CrystalReport1.LogOnInfo[0] = dsn=;uid=;pwd=bigboard;dsq=?
```

«Connects to a password-protected Paradox database. All that is being passed is the password bigboard.»

Remarks

- Use the LogOnInfo property when the report connects to multiple data sources that require different log on information. If the report connects to only a single ODBC data source or SQL server, and only a single set of log on information is required, simply pass 0 as the array index, or use *Connect*, *Page 498*, can be used instead.
- Use a separate line of code for each table for which you want to change the logon info.
- The order of tables in the array must conform to the order of tables in the report. (You can use the Database | Set Location command to determine the order of tables in the report.)
- The first table in the report is array index (0), the second file is (1), etc. For example, to change the logon information of the first and third tables in a report to the NEW server, use the following syntax:

```
CrystalReport1.LogOnInfo(0) = "DSN = NEW;UID = 734;PWD =  
bigboard;DSQ = Administration1"CrystalReport1.LogOnInfo(2) =  
"DSN = NEW;UID = 734;PWD = bigboard;DSQ = Administration2"
```

- LogOnInfo is an array property that is available at runtime only.
- Enter the parameters necessary to log on to each SQL server table that you need to change information for in your report. Parameters should be in the following format:

DSN = name;UID = userID;PWD = password;DSQ = database qualifier

- *name* is the server name,
- *userID* is the name you have been assigned for logging on to the SQL server,
- *password* is the password you have been assigned for logging on to the SQL server, and
- *database qualifier* is the database name if your server uses the database concept.

- The database qualifier parameter (DSQ) is required only when it is applicable to the ODBC / SQL driver you are using. If your DBMS does not use the database concept, you do not need to specify the DSQ parameter.
- Before you can use this property for an ODBC/SQL database, you must install the ODBC / SQL driver for whatever SQL database you are planning to use, and put the Database/BIN location in your path.
- If you are connecting to an SQL or other password protected database directly, without going through ODBC, use the name of the SQL server for the DSN parameter.
- *RetrieveLogonInfo*, *Page 604*, can be used to populate this property with log on information automatically.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

MarginBottom

Description

Sets the bottom margin for the specified report.

Usage

```
[form.]Report.MarginBottom[=MarginSetting%]
```

For example:

```
CrystalReport1.MarginBottom = 720
```

«Sets a 1/2 inch bottom margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips. A twip is 1/20th of a point. There are 72 points and thus 1440 twips in an inch.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetMargins, Page 305

MarginLeft

Description

Sets the left margin for the specified report.

Usage

```
[form.]Report.MarginLeft [=MarginSetting%]
```

For example:

```
CrystalReport1.MarginLeft = 1440
```

«Sets a 1 inch left margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetMargins, Page 305

MarginRight

Description

Sets the right margin for the specified report.

Usage

```
[form.]Report.MarginRight [=MarginSetting%]
```

For example:

```
CrystalReport1.MarginRight=1440  
«Sets a 1 inch right margin for the report (1 inch = 1440 twips).»
```

Remarks

MarginSetting is the margin you want, in twips.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetMargins, Page 305

MarginTop

Description

Sets the top margin for the specified report.

Usage

```
[form.]Report.MarginTop[=MarginSetting%]
```

For example:

```
CrystalReport1.MarginTop = 720  
«Sets a 1/2 inch top margin for the report (1 inch = 1440 twips).»
```

Remarks

MarginSetting is the margin you want, in twips.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetMargins, Page 305

ParameterFields

Description

Changes the default value of the specified parameter field. When the prompting dialog box appears for the parameter field, the value you specify with this property is the value you are prompted with.

Usage

```
[form.]Report.ParameterFields(ArrayIndex) [= "ParameterName;  
NewValue;SetCurrentValue"]
```

Remarks

- The parameter, SetCurrentValue can either be set to TRUE or FALSE.
- If set to TRUE, the parameter value is passed to the current value in the report; the user is not prompted to enter the parameter value.
- If set to FALSE, the parameter value is passed to the default value for the parameter; the user is prompted to enter the parameter value, with the value you set showing as the default value.
- The default value for SetCurrentValue is FALSE.
- This property does not eliminate the prompt by specifying a current value for the parameter field. You will still be prompted but the default value in the prompt will be the value you specify.
- Use a separate line of code for each parameter field for which you want to change the value.
- The order of values in the array must conform to the order of parameter fields in the report.
- The first parameter field in the report is array index (0), the second is (1), etc.
- For example, to change the value of the first parameter field in a report (parameter1) to “red” use the following syntax (user will not be prompted to enter a value):

```
CrystalReport1.ParameterFields(0) = "parameter1;red;TRUE"
```

- Or, to prompt the user to change the value of the third parameter field in a report (parameter3) use the following syntax (user will be prompted to use the default value set using the NewValue parameter below - “blue”):

```
CrystalReport1.ParameterFields(2) = "parameter3;blue;FALSE"
```

- This property/method is available for subreports.

Data Type

Array of strings

Availability

Runtime only

Related Report Engine Functions

PESetNthParameterField, Page 315

Password

Description

Enters the password needed to use database tables on a restricted Access .MDB file.

Usage

```
[form.]Report.Password [= Password$]
```

For example:

```
CrystalReport1.Password = "dogsncats"
```

«Enters the password *dogsncats*.»

Data Type

String

Availability

Runtime

Remarks

- Enter the password you have been assigned.
- Some Access databases have both database level security and user level security. If your database contains only database level security, simply assign your password to this property. If your database also has user level security, append ASCII character 10 to your database level password, followed by your user level password. For example:

```
CrystalReport1.Password = "dbpassword" + Chr(10) + "userpswd"
```

Related Report Engine Functions

PESetNthTableSessionInfo, Page 322

PrintDay

Description

Sets the day component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintDay[=Day%]
```

For example:

```
CrystalReport1.PrintDay = 23
```

«Sets 23 as the print day.»

Remarks

- Enter a value from 1 to 31.
- *PrintYear, Page 552, PrintMonth, Page 551*, and *PrintDay* work together to define the date that the report is to be printed. All three properties must be set in order to define a new print date. If all three properties are not set, the date saved with the report is used. This may be the user's default date if none has been specified in the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintDate, Page 324

PrinterCollation

Description

If you specify more than one copy to be printed (using the PrinterCopies property), PrinterCollation specifies whether or not the copies will be collated.

Usage

```
[form.]Report.PrinterCollation[=CollationCode%]
```

For example:

```
CrystalReport1.PrinterCollation = 1
```

«Collates the copies of the specified report.»

Remarks

Select your CollationCode% value from the following table:

Status	Code
Uncollated	0
Collated	1
Default Collation	2

This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer (Enumerated)

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintOptions, Page 326

PrinterCopies

Description

Sets the number of report copies to be printed.

Usage

```
[form.]Report.PrinterCopies[=NumCopies%]
```

For example:

```
CrystalReport1.PrinterCopies = 3
```

«Specifies that the program is to print three copies of the report.»

Remarks

- The number used for PrinterCopies must not be zero or a negative value.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintOptions, Page 326

PrinterDriver

Description

Sets the name of the printer driver that is to print the report.

Usage

```
[form.]Report.PrinterDriver [= DriverName$]
```

For example:

```
CrystalReport1.PrinterDriver = "Epson24.drv"
```

«Sets the printer driver to be the Epson 24 pin driver.»

Remarks

- PrinterDriver, *PrinterName*, *Page 537*, and *PrinterPort*, *Page 538*, work together to define the printer that the report is to be sent to. All three properties must be set in order to define a new printer. If all three properties are not set, the printer defined in the report will be used. This may be the user's default printer if none has been specified in the report.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

- For an example of how to use this property look in the WIN.INI file under the Devices section. You will find something like this:

```
[Devices]
HP LaserJet 4/4M=HPPCL5MS, hp4_tech_1
```

- The *PrinterName* is the HP LaserJet 4/4M, left of the = sign
- The *PrinterDriver* is the HPPCL5MS, first variable after the = sign
- The *PrinterPort* is the hp4_tech_1, the second variable after the = sign. Port is often something like "LPT1:"

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESelectPrinter, Page 267

PrinterName

Description

Sets the name of the printer that is to print the report.

Usage

```
[form.]Report.PrinterName[ = PrinterName$ ]
```

For example:

```
CrystalReport1.PrinterName = "Epson LQ-850"  
«Specifies the Epson LQ-850 printer.»
```

Remarks

- *PrinterDriver*, Page 536, *PrinterName*, and *PrinterPort*, Page 538, work together to define the printer that the report is to be sent to. All three properties must be set in order to define a new printer. If all three properties are not set, the printer defined in the report will be used. This may be the user's default printer if none has been specified in the report.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

- For an example of how to use this property look in the WIN.INI file under the Devices section. You will find something like this:

```
[Devices]  
HP LaserJet 4/4M=HPPCL5MS, hp4_tech_1
```

- The *PrinterName* is the HP LaserJet 4/4M, left of the = sign
- The *PrinterDriver* is the HPPCL5MS, first variable after the = sign
- The *PrinterPort* is the hp4_tech_1, the second variable after the = sign. Port is often something like "LPT1:"

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESelectPrinter, Page 267

PrinterPort

Description

Sets the name of the printer port for the specified printer.

Usage

```
[form.]Report.PrinterPort [= PortName$]
```

For example:

```
CrystalReport1.PrinterPort = "LPT1"
```

«Sets the printer port to LPT1.»

Remarks

- *PrinterDriver, Page 536, PrinterName, Page 537*, and PrinterPort work together to define the printer that the report is to be sent to. All three properties must be set in order to define a new printer. If all three properties are not set, the printer defined in the report will be used. This may be the user's default printer if none has been specified in the report.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

- For an example of how to use this property look in the WIN.INI file under the Devices section. You will find something like this:

```
[Devices]
HP LaserJet 4/4M=HPPCL5MS, hp4_tech_1
```

- The PrinterName is the HP LaserJet 4/4M, left of the = sign
- The PrinterDriver is the HPPCL5MS, first variable after the = sign
- The PrinterPort is the hp4_tech_1, the second variable after the = sign. Port is often something like "LPT1:"

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESelectPrinter, Page 267

PrinterStartPage

Description

Sets the first page to be printed.

Usage

```
[form.]Report.PrinterStartPage [= StartPage%]
```

For example:

```
CrystalReport1.PrinterStartPage = 7
```

«Specifies that printing is to begin with Page 7 of the report.»

Remarks

- If a value less than or equal to 0 is used for PrinterStartPage, the value is ignored and printing starts with Page 1.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintOptions, Page 326

PrinterStopPage

Description

Sets the last page to be printed.

Usage

```
[form.]Report.PrinterStopPage [=StopPage%]
```

For example:

```
CrystalReport1.PrinterStopPage = 12
```

«Specifies that the printing is to end with Page 12 of the report.»

Remarks

- Use a value of 0 for PrinterStopPage to indicate that printing is to continue through to the last page.
- This property/method is available for subreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintOptions, Page 326

PrintFileCharSepQuote

Description

Sets the quote character used to enclose alphanumeric field data when printing to a file using character-separated format.

Usage

```
[form.]Report.PrintFileCharSepQuote[ =Quote$ ]
```

For example:

```
CrystalReport1.PrintFileCharSepQuote = ""
```

«Uses the quotation character (') to surround values saved in character-separated format.»

Remarks

- Applies only when *PrintFileType*, *Page 547*, is 5 - Character-separated values.
- Applies only when *Destination*, *Page 502*, is 2 - File, 3 - E-mail to MAPI, or 4 - E-mail to VIM.
- If you assign a string to PrintFileCharSepQuote that is longer than one character, only the first character of that string is used. For example, if you assign “quote” to the property, only “q” is recognized.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*

PrintFileCharSepSeparator

Description

Sets the character(s) you want to use to separate the fields when printing to a file using the Character Separated Value format.

Usage

```
[form.]Report.PrintFileCharSepSeparator [=Separator$]
```

For example:

```
CrystalReport1.PrintFileCharSepSeparator= "@"
```

«Specifies that the "@" character is to be used for separating field values.»

Remarks

- Applies only when *PrintFileType*, *Page 547*, is 5 - Character-separated values.
- Applies only when *Destination*, *Page 502*, is 2 - File, 3 - E-mail to MAPI, or 4 - E-mail to VIM.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*

PrintFileLinesPerPage

Description

Indicates the number of lines to be printed before the page break. The default is 60 lines.

Usage

```
[form.]Report.PrintFileLinesPerPage
```

For example:

```
CrystalReport1.PrintFileLinesPerPage = 50
```

«Fifty lines will be printed before a page break.»

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Runtime

Remarks

This property/method is available for subreports.

PrintFileName

Description

Specifies the name of the file to which the report is to be printed.

Usage

```
[form.]Report.PrintFileName [= FileName$]
```

For example:

```
CrystalReport1.PrintFileName = "c:\crw\cust_rpt.txt"
```

«Prints the report to a file named "cust_rpt.txt" in the C:\CRW directory.»

Remarks

- You can double-click this property or click the ellipsis (...) in the Properties box to call up the Choose Print Filename dialog box. In that dialog box, select the name of the file and the path to which you want the program to print the report.

- Select a value for this property only if you are printing to a file, if the value you assigned to the *Destination*, *Page 502*, is 2 - File.
- This property/method is available for subreports.

NOTE: If you want to specify the *PrintFileName* at runtime, make certain that you enclose it in quotes in your code.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*

PrintFileODBCPassword

Description

Used whenever you want to export in ODBC format, specifies the password that you need to connect to the data source.

Usage

```
[form.]Report.PrintFileODBCPassword[ = Password$ ]
```

For example:

```
CrystalReport1.PrintFileODBCPassword = "merry%%5"
```

«"merry%%5" is the name of the password to connect to the data source.»

Remarks

- This is only required if the ODBC datasource that you are exporting to requires a password.
- This property/method is available for subreports.

Data Type

String

Availability

Design Time; Runtime

PrintFileODBCSource

Description

Used whenever you export in ODBC format. Specifies the name of the data source that you want to export to.

Usage

```
[form.]Report.PrintFileODBCSource[ = DataSource$ ]
```

For example:

```
CrystalReport1.PrintFileODBCSource = "pickle"
```

« "pickle" is the name of the data source that you want to export to.»

Data Type

String

Availability

Design Time; Runtime

Remarks

This property/method is available for subreports.

PrintFileODBCTable

Description

Used whenever you want to export in ODBC format, specifies the name of the table you want to export to in the data source.

Usage

```
[form.]Report.PrintFileODBCTable[= TableName$]
```

For example:

```
CrystalReport1.PrintFileODBCTable = "Employees"
```

«"Employees" is the name of the table in the data source to export to.»

Data Type

String

Availability

Design Time; Runtime

Remarks

This property/method is available for subreports.

PrintFileODBCUser

Description

Used whenever you export in ODBC format, this specifies the User ID that you need to connect to the data source

Usage

```
[form.]Report.PrintFileODBCUser[= UserID$]
```

For example:

```
CrystalReport1.PrintFileODBCUser = "LisaB"
```

«"LisaB" is the User ID needed to connect to the data source.»

Data Type

String

Availability

Design Time; Runtime

Remarks

This property/method is available for subreports.

PrintFileType

Description

Specifies the type of print file used when printing a report to a file.

Usage

```
[form.]Report.PrintFileType[ = FileType%]
```

For example:

```
CrystalReport1.PrintFileType = 1
```

«Prints the report to a file in a tab separated format.»

Remarks

Select one of the following print file types if you are printing to a file, if the value you assigned to the *Destination*, *Page 502*, is 2 - File.

0 - Record

Record style (columns of values). Does not use commas or separators. Outputs every record with a fixed field width.

1 - Tab separated

Tab separated values. Presents data in tabular form. Encloses alphanumeric field data in quotes and separates fields with tabs.

2 - Text

Text style. Saves the data in ASCII text format with all values separated by spaces. This style looks most like the printed page.

3 - DIF

Saves the data in DIF (data interchange format) format. This format is often used for the transfer of data between different spreadsheet programs.

4 - CSV

Comma separated values. Encloses alphanumeric field data in quotes and separates fields with commas.

5 - Character Separated

Saves the data as character separated values in ASCII text format. All values are separated by a character or characters specified by the *PrintFileCharSepSeparator*, *Page 542*.

6 - Tab separated text

Saves the data in ASCII text format with all values separated by tabs.

7 - Seagate Crystal Reports (RPT)

Standard Seagate Crystal Reports (RPT) format is used. Most often used for sending the report to another user via e-mail.

8 - Excel 2.1 XLS

Exports the report as a Microsoft Excel 2.1 worksheet.

9 - Excel 3.0 XLS

Exports the report as a Microsoft Excel 3.0 worksheet.

10 - Excel 4.0 XLS

Exports the report as a Microsoft Excel 4.0 worksheet.

11 - Lotus 1-2-3 WK1

Exports the report as a Lotus 1-2-3 WK1 format worksheet.

12 - Lotus 1-2-3 WK3

Exports the report as a Lotus 1-2-3 WK3 format worksheet.

13 - Lotus 1-2-3 WKS

Exports the report as a Lotus 1-2-3 WKS format worksheet.

14 - Quattro Pro 5.0 WB1 (16-bit only)

Exports the report as a Quattro Pro 5.0 WB1 format file.

15 - RTF

Saves the data in Rich Text Format.

16 - Word for DOS (16-bit only)

Uses the Microsoft Word for DOS format to save the data in the report.

17 - Word for Windows

Uses the Microsoft Word for Windows format to save the data in the report.

18 - WordPerfect (16-bit only)

Uses WordPerfect format to save the data in the report.

19 - Excel 5

Uses Excel 5 format to save the data in the report.

20 - HTML 3

Uses HTML 3 format to save the data in the report.

21 - HTML Internet Explorer

Uses the Internet Explorer version of HTML to save the data in the report.

22 - HTML Netscape

Uses the Netscape version of HTML to save the data in the report.

23 - HTML Netscape

Saves the data in ASCII text format, broken into pages.

24 - ODBC

Exports to a database format corresponding with an ODBC data source that you specify.

NOTE: If you specify a table name for PrintFileODBCTable that already exists in the database, you will receive an error stating that the table already exists.

25 - Paginated Text

Text style. Saves the data in ASCII text format with pagination information.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

This property/method is available for subreports.

Data Type

Integer (Enumerated)

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, Page 146

PrintFileUseRptDateFmt

Description

When printing to a file, indicates whether or not the program should save dates in the same date format (MDY, DMY, etc.) that is used in the report or instead, optimize the dates for the file format you have selected.

Usage

```
[form.]Report.PrintFileUseRptDateFmt [= TrueFalse%]
```

For example:

```
CrystalReport1.PrintFileUseRptDateFmt = 1
```

«Specifies that the program should print dates in the same format as used in the report.»

Remarks

- Applies only when *PrintFileType*, *Page 547*, is 0 - Record, 1 - Tab-separated, 3 - Data Interchange Format (DIF), 4 - CSV, or 5 - Character-Separated.
- Applies only when *Destination*, *Page 502*, is 2 - File, 3 - E-mail to MAPI, or 4 - E-mail to VIM.
- For *TrueFalse%*, use one of the following values:
 - False = 0
 - True = 1
- This property/method is available for subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*

PrintFileUseRptNumberFmt

Description

When printing to a file, indicates whether or not the program should print numbers in the same format (decimal places, negatives, etc.) that you have used in the report or instead, optimize the numbers for the file format you have selected.

Usage

```
[form.]Report.PrintFileUseRptNumberFmt [=TrueFalse%]
```

For example:

```
CrystalReport1.PrintFileUseRptNumberFmt = 1
```

«Specifies that the program should print numbers in the same format as used in the report.»

Remarks

- Applies only when *PrintFileType*, *Page 547*, is 0 - Record, 1 - Tab-separated, 3 - Data Interchange Format (DIF), 4 - CSV, or 5 - Character-Separated.
- Applies only when *Destination*, *Page 502*, is 2 - File, 3 - E-mail to MAPI, or 4 - E-mail to VIM.
- For *TrueFalse%*, use one of the following values:
 - False = 0
 - True = 1
- This property/method is available for subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*

PrintMonth

Description

Sets the month component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintMonth[= Month%]
```

For example:

```
CrystalReport1.PrintMonth= 7
```

«Sets July as the print month.»

Remarks

- Enter a value from 1-12 where January = 1, and December = 12.
- *PrintYear*, *Page 552*, *PrintMonth*, and *PrintDay*, *Page 533*, work together to define the date that the report is to be printed. All three properties must be set in order to define a new print date. If all three properties are not set, the date saved with the report is used. This may be the user's default date if none has been specified in the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintDate, *Page 324*

PrintYear

Description

Sets the year component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintYear[=Year%]
```

For example:

```
CrystalReport1.PrintYear = 1994
```

«Sets the year component of the print date to 1994.»

Remarks

- Enter the print year as a four-digit number.
- *PrintYear*, *PrintMonth*, *Page 551*, and *PrintDay*, *Page 533*, function together. You must change the value of all three to change the print date. If you do not change all three, the print date saved with the report is used. This may be the current date if a specific date is not saved with the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PESetPrintDate, Page 324

ProgressDialog

Description

Enables/disables the display of the Progress dialog box. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

Usage

```
[form.]ReportProgressDialog [= {True | False}]
```

For example:

```
CrystalReport1ProgressDialog = False
```

«Turns off the Progress dialog box that usually appears during exporting or printing.»

Remarks

- Use this property to indicate whether or not a progress dialog box should be displayed while the report is printed or exported. This property is set to True by default.
- This property/method is available for subreports.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PEEnableProgressDialog, Page 142

RecordsPrinted

Description

Determines the number of records actually printed.

Usage

```
[form.]Report.RecordsPrinted
```

For example:

```
Printed& = CrystalReport1.RecordsPrinted
```

«Fetches the number of records printed and stores it in the *Printed* variable.»

Remarks

- If the report being printed contains one or more group selection formulas, the value returned by RecordsPrinted may be significantly less than the value returned by RecordsSelected. Otherwise, this value should equal *RecordsSelected, Page 555*.
- This property/method is available for subreports.

Data Type

Long

Availability

Runtime

Related Report Engine Functions

PEGetJobStatus, Page 171

RecordsRead

Description

Determines the number of records actually processed.

Usage

```
[form.]Report.RecordsRead
```

For example:

```
Read% = CrystalReport1.RecordsRead
```

«Fetches the number of records read and saves it in the *Read* variable.»

Remarks

- If the Crystal Report Engine generates a SQL query to obtain data from a SQL database when the report is printed, RecordsRead will only return the number of records received by the Crystal Report Engine from the query. This value may be significantly smaller than the number of records actually in the SQL database table.
- This property/method is available for subreports.

Data Type

Long

Availability

Runtime

Related Report Engine Functions

PEGetJobStatus, Page 171

RecordsSelected

Description

Determines the number of records selected for inclusion in the report out of the total number of records read.

Usage

```
[form.]Report.RecordsSelected
```

For example:

```
Selected& = CrystalReport1.RecordsSelected
```

«Fetches the number of records selected and saves it in the *Selected* variable.»

Remarks

- RecordsSelected will return a value anywhere between zero and the value returned by *RecordsRead*, *Page 555*. The value returned by RecordsSelected depends on the queries and selection formulas set up in the report.
- This property/method is available for subreports.

Data Type

Long

Availability

Runtime

Related Report Engine Functions

PEGetJobStatus, *Page 171*

ReportDisplayPage

Description

Indicates which page of a multi-page report is currently being displayed in the preview window.

Usage

```
[form.]Report.ReportDisplayPage
```

For example:

```
Result% = CrystalReport1.ReportDisplayPage
```

«Fetches the number of the displayed page and stores it in the *Result* variable.»

Data Type

Integer

Availability

Runtime

Remarks

This property/method is available for subreports.

Related Report Engine Functions

PEGetJobStatus, Page 171

ReportFileName

Description

Specifies the report to be printed.

Usage

```
[ form. ]Report.ReportFileName[ = ReportName$ ]
```

For example:

```
CrystalReport1.ReportFileName = "c:\crw\company.rpt"
```

«Prints the report named “company.rpt” that is located in the C:\CRW directory.»

Remarks

- You can double-click this property or click the ellipsis (...) in the Properties box to call up the Choose Report File dialog box. In that dialog box, select the name and path of the report you want the program to print in response to a Crystal ActiveX Control event.
- This property/method is available for subreports.

NOTE: If you want to specify the ReportFileName at runtime, make certain that you enclose it in quotes in your code.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEOpenPrintJob, Page 248

ReportLatestPage

Description

Determines the last page printed in the specified report.

NOTE: ReportLatestPage will only contain the last page number after calling CrystalReport1.PageCount.

Usage

```
[form.]Report.ReportLatestPage
```

For example:

```
Latest% = CrystalReport1.ReportLatestPage
```

«Fetches the number of the last page printed and stores it in the *Latest* variable.»

Data Type

Integer

Availability

Runtime

Remarks

This property/method is available for subreports.

Related Report Engine Functions

PEGetJobStatus, Page 171

ReportSource

Description

Specifies the source of the report as a report file, a Visual Basic data control, or a True Grid data control.

Usage

```
[form.]Report.ReportSource
```

For example:

```
CrystalReport1.ReportSource = 1
```

«Specifies the report source as the TrueDBGrid control.»

Data Type

Integer

Availability

Design Time and Runtime

Remarks

This property/method is available for subreports.

ReportStartPage

Description

Determines the first page printed in the specified report.

Usage

```
[form.]Report.ReportStartPage
```

For example:

```
StartPage% = CrystalReport1.ReportStartPage
```

«Fetches the number of the first page printed and stores it in the *StartPage* variable.»

Data Type

Integer

Availability

Runtime

Remarks

This property/method is available for subreports.

Related Report Engine Functions

PEGetJobStatus, Page 171

ReportTitle

Description

Specifies a title for the report.

Usage

```
[form.]Report.ReportTitle[= rptTitle$]
```

For example:

```
CrystalReport1.ReportTitle = "My Report"
```

«Applies the title "My Report" to the report.»

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESetReportSummaryInfo, Page 327

SectionFont

Description

Sets the font for one or more sections in the specified report.

Usage

```
[form.]Report.SectionFont(SequentialIndex%) [= sectionCode; fontName;  
size; italic; bold;underline;strikethru$]
```

For example:

```
CrystalReport1.SectionFont(0) = "Footer;Arial;12;N;N;N;Y"
```

«Sets the font for the footer section to 12 point, Arial, strikethrough.»

Remarks

- With SectionFont, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make the “Action=1” call.
- The array index value for SectionFont simply specifies the sequence number for the change. Thus:

```
CrystalReport1.SectionFont(0) = "DETAIL;Arial;12;N;N;N;Y"
```

when making changes to the DETAIL section only, but

```
CrystalReport1.SectionFont(0) = "HEADER;Arial;12;N;N;N;Y"  
CrystalReport1.SectionFont(1) = "DETAIL;Arial;12;N;N;N;Y"
```

when making changes to more than one section.

Use the following table as a guide in supplying the required values for this property:

Parameter	Data type	Value expected
sectionCode	string	Please refer to the <i>Section Codes (32-bit), Page 494</i> .
fontName	string	The actual font name (i.e., Arial or Helvetica).
size	number	The size of the font in points (i.e., 12 or 16).
italic	character	T = true, F = False, X = as is*
bold	character	T = true, F = False, X = as is*
underline	character	T = true, F = False, X = as is*

<i>Parameter</i>	<i>Data type</i>	<i>Value expected</i>
strikethrough	character	T = true, F = False, X = as is*

*X (as is) uses the value saved with the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetFont, Page 282

SectionFormat

Description

Sets the format for one or more sections in the specified report.

Usage

```
[form.]Report.SectionFormat(SectionArrayIndex%) [= sectionCode;
visible; newPageBefore; newPageAfter; keepTogether;
SuppressBlankSection; resetPageNAfter; printAtBottomOfPage;
underlaySection; backgroundColor]
```

For example:

```
CrystalReport1.SectionFormat(0)= "GH2;F;X;X;X;X;X;X;255.0.0"
```

«Hides the Group Header 2 section (visible = F) and changes the background color to red while maintaining default settings for all other switches.»

Remarks

With SectionFormat, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make the “Action=1” call. The sequential index value for SectionFormat simply specifies the sequence number for the change. Thus:

```
CrystalReport1.SectionFormat(0) = "DETAIL;T;F;F;X;X;X;X;255.0.0"
```

when making changes to the DETAIL section only, but

```
CrystalReport1.SectionFormat(0) = "HEADER;T;F;F;X;X;X;X;255.0.0"
```

```
CrystalReport1.SectionFormat(1) = "DETAIL;T;F;F;X;X;X;X;255.0.0"
```

when making changes to more than one section.

Use the following table as a reference when entering parameter values for this property:

Parameter	Expected value
sectionCode	Please refer to the <i>Section Codes (32-bit), Page 494</i> .
Visible	T = true, F = False, X = as is*
newPageBefore	T = true, F = False, X = as is*
newPageAfter	T = true, F = False, X = as is*
keepTogether	T = true, F = False, X = as is*
suppressBlank	T = true, F = False, X = as is*
resetPageNAfter	T = true, F = False, X = as is*
printAtPageBottom	T = true, F = False, X = as is*
underlaySection	T = true, F = False, X = as is*
backgroundColor	Supply a RGB (Red, Green, Blue) value in the following format: <R>.<G>. where R, G, and B are each integers with a range from 0 to 255. For example: 189.210.100. If you do not want to change the color, do not place anything in this parameter.

* X (as is) uses the settings saved with the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

Related Report Engine Functions

PESetSectionFormat, Page 331

SectionLineHeight

Description

Specifies the line height in twips. A twip is 1/1440 inch; there are 20 twips in a point.

Usage

```
[form.]Report.SectionLineHeight(SequentialIndex%) [=sectionCode; line;  
height; ascent$]
```

For example:

```
CrystalReport1.SectionLineHeight(0) = "GH0; 1; 500; 300"
```

«Sets the line height for the second line in the group header zero section to a height of 500 twips with an ascent of 300 twips.»

Remarks

With SectionLineHeight, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make the “Action=1” call. The sequential index value for SectionLineHeight simply specifies the sequence number for the change. Thus:

```
CrystalReport1.SectionLineHeight(0)= "DETAIL;1;500;300"
```

when making changes to the DETAIL section only, but

```
CrystalReport1.SectionLineHeight(0) = "HEADER;1;500;300"  
CrystalReport1.SectionLineHeight(1) = "DETAIL;1;500;300"
```

when making changes to more than one section.

Use the following table as a guide in supplying the required values for this property:

Parameter	Explanation
sectionCode	Specifies the section code for the report section(s) for which you want to set a new line height. See <i>Section Codes (32-bit)</i> , Page 494.
lineN	Specifies the line(s) for which you want to set the line height. Line numbers in a section are 0 origin: the first line number is 0, the second is 1, etc.
height	Specifies the line height in twips. A twip is 1/1440 inch; there are 20 twips in a point.

<i>Parameter</i>	<i>Explanation</i>
ascent	Specifies the ascent in twips. Ascent is the distance from the top of the allotted line space (line height) to the baseline of the font. The ascent parameter is used to specify the position of the baseline if you specify an oversized or undersized line height. If you set ascent to 0, the program puts the baseline at the top of the space; if you set ascent to the same value as height, the program sets the baseline at the bottom of the space. For any other baseline, specify the ascent in twips.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

SectionMinHeight

Description

Sets the minimum section height for the specified report section.

Usage

```
[form.]Report.SectionMinHeight  
(SequentialIndex%) [=sectionCode:minHeight$]
```

For example:

```
CrystalReport1.SectionMinHeight(0) = "ALL;500"  
«Sets the minimum height for all sections to 500 twips.»
```

Remarks

- With SectionMinHeight, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make the “Action=1” call.
- The array index value for SectionMinHeight simply specifies the sequence number for the change. Thus:

```
CrystalReport1.SectionMinHeight(0) = "DETAIL;500"
```

when making changes to the DETAIL section only, but

```
CrystalReport1.SectionMinHeight(0) = "HEADER;500"
```

```
CrystalReport1.SectionMinHeight(1) = "DETAIL;500"
```

when making changes to more than one section.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetMinimumSectionHeight, Page 307

SelectionFormula

Description

Specifies the records to be used when printing the report.

Usage

```
[form.]Report.SelectionFormula [= SelectionFormula$]
```

For example:

```
CrystalReport1.SelectionFormula = "{file.QTY} > 5"
```

«Include only those records that have a quantity greater than 5 in the {file.Qty} field.»

Remarks

- Enter the selection formula just as you would enter it in the Formula Editor.
- Make certain that you enclose your selection formula in double quotes.
- If your selection formula includes internal quotes, for example:

```
{file.STATE} = "CA"
```

change all of the internal double quotes to single quotes and then surround the entire selection formula in double quotes as follows:

```
"{file.STATE} = 'CA'"
```

- If you have created a selection formula in your report at Design Time, any selection formula you enter here will be appended to that selection formula. Thus, your records will be selected based on a combination of the two selection formulas.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PESetSelectionFormula, Page 336

SessionHandle

Description

Sets the session handle for a user once the UserName and Password properties have opened an Access.mdb file for use by the report.

Usage

```
[form.]Report.SessionHandle [= Handle%]
```

Remarks

If you have already opened a Jet session in your Visual Basic application, you can set this property to be the current session handle. Otherwise, you will have to use the Password and UserName properties to establish the Jet session.

For example:

```
CrystalReport1.SessionHandle = CurrentSessionHandle
```

«Sets the session handle to the session handle returned elsewhere in the application and stored in the variable *CurrentSessionHandle*.»

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PESetNthTableSessionInfo, Page 322

SortFields

Description

Specifies the field(s) that are to be used to sort your data when the report is printed.

Usage

```
[form.]Report.SortFields(ArrayIndex)[= "+|-| SortField"]
```

Enter the fields on which you want the data in your report to be sorted.

For example, to sort an order database alphabetically by customer, and then by order date, you can enter code similar to the following:

```
CrystalReport1.SortFields(0) = "+{orders.CUSTOMER}"
CrystalReport1.SortFields(1) = "+{orders.ORDERDATE}"
```

Remarks

- SortFields is an array property available only at runtime.
- Use a separate line of code to specify each sort field.
- Enter sort fields in the order that you want them to sort your report. For example, if you want your report to be sorted first on field A and then on field B, specify sort field A in your first line of code and sort field B in your second line of code.
- The sort field you specify must be assigned array index 0, the second sort field must be assigned array index 1, etc.
- The index values you assign must be continuous; no gaps are allowed (0, 1, 2 = OK, 0, 1, 3 = wrong).

- Array index values must be subscripted in the code immediately after the property name (i.e., CrystalReport1.SortFields(0) =).
- If you have specified sort fields for your report at Design Time, any sort fields you enter here will replace the sort fields in your report.
- If you do not use this property, the program will use the sorting instructions that you specified in the report.
- If you want to clear the sort fields in your report, use an empty string (i.e., CrystalReport1.SortFields(0) = "").
- Enclose field names in braces.
- Sort fields can be database fields or formula fields. If you sort on a formula field, use the @ sign before the formula name (i.e., {@FORMULANAME}).

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetNthSortField, Page 316

SQLQuery

Description

Sets the SQL query string used by the specified report.

Usage

```
[form.]Report.SQLQuery[=SQLQuery$]
```

For example:

```
CrystalReport1.SQLQuery = "SELECT authors.au_id, authors.au_lname,
authors.au_fname FROM pubs2.dbo.authors authors WHERE authors.au_lname
> 'Madison'"
```

«Queries the SQL database to return only the records where the authors last name falls after Madison alphabetically.»

Remarks

- You may only change the WHERE, FROM, and ORDER BY sections of an SQL query. Although the property requires that you enter the entire SQL query, the SELECT section must not be different from the original query in the report.
- To change the ORDER BY clause, you must place a carriage return and linefeed characters after the WHERE clause and before the ORDER BY clause in this way:

```
CrystalReport1.SQLQuery="SELECT...FROM...
WHERE..." + CHR$(13) + CHR$(10) + "ORDER BY..."
```

- This property is active only if you are using an SQL or ODBC data source in your report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEGetSQLQuery, Page 226

PESetSQLQuery, Page 337

Status

Description

Determines the print status for the specified report.

Usage

```
[form.]Report.Status
```

For example:

```
Status% = CrystalReport1.Status  
«Fetches the print status and saves it to the Status variable.»
```

Remarks

The Status property will return one of the following values:

- 0 - The report has not been printed or has not finished printing.
- 3 - The report has finished printing.
- 5 - The job has been cancelled by the user.

This property/method is available for subreports.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEGetJobStatus, Page 171

StoredProcParam

Description

Sets the stored procedure parameters when using a report based on SQL stored procedures.

Usage

```
[form.]Report.StoredProcParam(Parameter Array Index%) [= newParameter$]
```

For example:

```
CrystalReport1.StoredProcParam(0)="06/14/1989"  
«Sets the first stored procedure parameter to the date June 14, 1989.»
```

Remarks

StoredProcParam sets the value of the specified parameter in an SQL database table that is based on a stored procedure. Pass the value you wish to set the parameter to as a string. If the parameter expects a different data type, you still must pass the value as a string. For example, to pass the integer value 396, use the string "396". The Crystal Report Engine will handle converting the value into integer format.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

Related Report Engine Functions

PESetNthParam, Page 313

SubreportToChange

Description

Specifies whether changes to any of several properties (see list in Remarks below) affect the main report (if you pass an empty string [""]) or a subreport (if you pass the name of the subreport).

Usage

```
[form.]Report.SubreportToChange (= SubreportName$)
```

For example:

```
CrystalReport1.SubreportToChange = ""  
«Changes to any of the properties apply to the main report.»  
CrystalReport1.SubreportToChange = "Subrpt2"  
«Changes to any of the properties apply to the Subrpt2 subreport»
```

Remarks

The following properties are affected by this property:

- Connect
 - DataFiles
 - DetailCopies
 - Formulas
 - GraphData
 - GraphOptions
 - GraphText
 - GraphType
 - GroupCondition
 - GroupSelectionFormula
 - GroupSortFields
 - LogonInfo
 - MarginBottom
 - MarginLeft
 - MarginRight
 - MarginTop
 - Password
 - PrintDay
 - PrintMonth
 - PrintYear
 - ReportTitle
 - SectionFont
 - SectionFormat
 - SectionLineHeight
 - SectionMinHeight
 - SelectionFormula
 - SessionHandle
 - SortFields
 - SQLQuery
 - StoredProcParam
 - UserName
- If the SubreportToChange property is set to:
 - the empty string, changing any of these properties affects the main report.
 - the name of a subreport, changing any of these properties affects the subreport
 - When you change the value of the SubreportToChange property, the current value of the properties in the list are updated to be what has been set for the selected subreport.
 - Calling the ReplaceSelectionFormula method will apply the new selection formula to the currently selected subreport.

NOTE: These properties only reflect values set by the programmer; they do not get values from the report.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Runtime

UserName

Description

Enters the name given to a user for logging on to a protected Access.mdb file to obtain data files needed by the report.

Usage

```
[form.]Report.UserName[ = Name$ ]
```

For example:

```
CrystalReport1.UserName = "MIS"
```

```
«Enters the user name "MIS".»
```

Remarks

- Enter the name you have been assigned.
- The name must be enclosed in quotes if the variable is being assigned at runtime.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Runtime

Related Report Engine Functions

PESetNthTableSessionInfo, Page 322

WindowAllowDrillDown

Description

Indicates whether or not drill-down on summary values is allowed in the preview window.

Usage

```
[form.]Report.WindowAllowDrillDown [= {True|False}]
```

For example:

```
CrystalReport1.WindowAllowDrillDown = FALSE
```

«Drill-down is not allowed in the preview window.»

Remarks

This property is set to False by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowBorderStyle

Description

Specifies the type of border for the preview window.

Usage

```
[form.]Report.WindowBorderStyle [= BorderStyle%]
```

For example:

```
CrystalReport1.WindowBorderStyle = 2
```

«Sets a sizable border style (Style #2) for the preview window.»

Remarks

Select one of the following border styles for the preview window:

- 0 = None (creates a window with no border).

- 1 = Fixed Single (creates a window of a fixed size with a single line border).
- 2 = Sizeable (creates a window that can be resized by the user).
- 3 = Fixed Double (creates a window of fixed size with a double line border).

Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer (Enumerated)

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowControlBox

Description

Specifies whether or not the preview window is to have a control (system menu) box in the upper left hand corner when the report is printed to a window.

Usage

```
[form.]Report.WindowControlBox [= {True|False}]
```

For example:

```
CrystalReport1.WindowControlBox = True
```

«Specifies that a control box (system menu) is to appear in the preview window.»

Remarks

- Select True if you want the window to contain a control box. Select False if you do not.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PENewOutputToWindow, Page 257

WindowControls

Description

Specifies whether or not the print controls are to appear in the preview window when printing a report to a window.

Usage

```
[form.]Report.WindowControls[={True|False}]
```

For example:

```
CrystalReport1.WindowControls = True
```

«Specifies that print controls are to appear in the preview window.»

Remarks

- Select True if you want the print controls to appear in the preview window.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEShowPrintControls, Page 346

WindowHeight

Description

Sets the height of the preview window when the report is printed to a window.

Usage

```
[form.]Report.WindowHeight [= Height%]
```

For example;

```
CrystalReport1.WindowHeight = 300
```

«Sets the height of the preview window to 300 pixels.»

Remarks

- If you are not satisfied with the default settings, enter the external height you want for your preview window in pixels. (A standard VGA monitor is 640 x 480 pixels.)
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowLeft

Description

Sets the distance, in pixels, that the preview window is to appear from the left edge of the parent window. If the preview window is a top level window, then the distance is measured from the left edge of the screen. (A standard VGA monitor is 640 x 480 pixels.)

Usage

```
[form.]Report.WindowLeft[= Distance%]
```

For example:

```
CrystalReport1.WindowLeft = 100
```

«Sets the left edge of the preview window 100 pixels from the left edge of the screen.»

Remarks

- If you are not satisfied with the default settings, enter the number of pixels you want between the left edge of the screen and the left edge of your window.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowMaxButton

Description

Specifies whether or not the preview window is to have an active, hidden, or grayed out maximize button when the report is printed to a window.

Usage

```
[form.]Report.WindowMaxButton[= {True|False}]
```

For example:

```
CrystalReport1.WindowMaxButton = False
```

«Specifies that no Maximize button is to appear in the preview window (in Windows 95, NT4+).
Specifies that the Maximize button will appear grayed out in the preview window (in Windows 3.x)»

Remarks

- Select True if you want the window to contain a maximize button. Select False if you do not.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.
- If you set both *WindowMaxButton* and *WindowMinButton, Page 580*, to False, the buttons will not appear at all (be hidden).
- If you set one of *WindowMaxButton* or *WindowMinButton* to True, the other button will appear grayed out.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowMinButton

Description

Specifies whether or not the preview window is to have an active, hidden, or grayed out minimize button when the report is printed to a window.

Usage

```
[form.]Report.WindowMinButton[ = {True|False}]
```

For example:

```
CrystalReport1.WindowMinButton = True
```

«Specifies that a Minimize button is to appear in the preview window (in Windows 95, NT4+).
Specifies that the Minimize button will be active (no grayed out) in the preview window (in Windows 3.x).»

Remarks

- Select True if you want the window to contain a minimize button. Select False if you do not.

- Select a value here only if you are printing to a window, if *Destination*, *Page 502*, is set to 0.
- If you set both *WindowMaxButton*, *Page 579*, and *WindowMinButton* to False, the buttons will not appear at all (be hidden).
- If you set one of *WindowMaxButton* or *WindowMinButton* to True, the other button will appear grayed out.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, *Page 257*

WindowParentHandle

Description

Specifies the handle of the parent window if the preview window is to be the child of another window.

Usage

```
[form.]Report.WindowParentHandle[ = ParentHandle%]
```

For example:

```
CrystalReport1.WindowParentHandle = Form1.hWnd
```

«Sets the WindowParentHandle to the handle of Form1. This specifies that the preview window is to be a child of Form1.»

Remarks

This is a runtime-only property.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEDisplayToWindow, Page 257

WindowShowCancelBtn

Description

Indicates whether or not a Cancel button is available in the preview window.

Usage

```
[form.]Report.WindowShowCancelBtn[ = {True|False}]
```

For example:

```
CrystalReport1.WindowShowCancelBtn = False
```

«No Cancel button is displayed in the preview window.»

Remarks

This property is set to False by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowCloseBtn

Description

Indicates whether or not a Close button is available in the preview window.

Usage

```
[form.]Report.WindowShowCloseBtn [= {True|False}]
```

For example:

```
CrystalReport1.WindowShowCloseBtn = True
```

«A Close button will appear in the preview window.»

Remarks

This property is set to False by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowExportBtn

Description

Indicates whether or not an Export button is available in the preview window.

Usage

```
[form.]Report.WindowShowExportBtn [= {True|False}]
```

For example:

```
CrystalReport1.WindowShowExportBtn = True
```

«The Export button appear in the preview window.»

Remarks

This property is set to True by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowGroupTree

Description

Indicates whether or not a Group Tree is displayed in the preview window.

Usage

```
[form.]Report.WindowShowGroupTree[ = {True|False}]
```

For example:

```
CrystalReport1.WindowShowGroupTree = False
```

«No Group Tree is displayed in the preview window.»

Remarks

This property is set to False by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowNavigationCtls

Description

Indicates whether or not the Navigation Controls are available in the preview window.

Usage

```
[form.]Report.WindowShowNavigationCtls [= {True|False}]
```

For example:

```
CrystalReport1.WindowShowNavigationCtls = True
```

«The preview window contains Navigation Controls.»

Remarks

This property is set to True by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowPrintBtn

Description

Indicates whether or not a Print button is available in the preview window.

Usage

```
[form.]Report.WindowShowPrintBtn[= {True|False}]
```

For example:

```
CrystalReport1.WindowShowPrintBtn = False
```

«No Print button is available in the preview window.»

Remarks

This property is set to True by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowPrintSetupBtn

Description

Indicates whether or not a Print Setup button is available in the preview window.

Usage

```
[form.]Report.WindowShowPrintSetupBtn[= {True|False}]
```

For example:

```
CrystalReport1.WindowShowPrintSetupBtn = False  
«No Print Setup button is available in the preview window.»
```

Remarks

- This property is set to False by default.
- The Print Setup button displays the Print Setup dialog box, allowing users to make changes to printer settings before printing a report that appears in the preview window.
- If the Print Setup button is disable, and the Print button is enabled (see *WindowShowPrintBtn*, *Page 586*), when a user clicks the Print button, the report will be sent to the printer selected when the report was created. If no printer was selected, or if the printer is unavailable, the report will be sent to the default printer for the current machine.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, *Page 341*

WindowShowProgressCtl

Description

Determines whether or not controls indicating the progress of a report being generated are displayed in the preview window.

Usage

```
[form.]Report.WindowShowProgressCtl[= {True|False}]
```

For example:

```
CrystalReport1.WindowShowProgressCtl = False  
«No Progress Controls are displayed in the preview window.»
```

Remarks

This property is set to True by default.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowRefreshBtn

Description

Indicates whether or not a Refresh button is available in the preview window.

Usage

```
[form.]Report.WindowShowRefreshBtn [= {True|False}]
```

For example:

```
CrystalReport1.WindowShowRefreshBtn = False
```

«No Refresh button is displayed in the preview window.»

Remarks

- This property is set to False by default.
- The Refresh button allows a user to refresh the data displayed in a report. The most current data can be obtained by refreshing report data, but the process of accessing the database containing the data can be time consuming and may burden system and network resources.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowSearchBtn

Description

Indicates whether or not a Search button is available in the preview window.

Usage

```
[form.]Report.WindowShowSearchBtn [= {True|False}]
```

For example:

```
CrystalReport1.WindowShowSearchBtn = True
```

«The Search button is displayed in the preview window.»

Remarks

- This property is set to False by default.
- The Search button allows a user to search for a specific field value in the report.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowShowZoomCtl

Description

Indicates whether or not Zoom Controls are available in the preview window.

Usage

```
[form.]Report.WindowShowZoomCtl[= {True|False}]
```

For example:

```
CrystalReport1.WindowShowZoomCtl = True
```

«Zoom Controls are displayed in the preview window.»

Remarks

- This property is set to True by default.
- Zoom controls allow a user to zoom in or out on report data, enlarging or reducing the report on screen.

Data Type

Boolean

Availability

Design Time; Runtime

Related Report Engine Functions

PESetWindowOptions, Page 341

WindowState

Description

Sets the state of the preview window, normal, minimized, or maximized, when the report is printed.

Usage

```
[form.]Report.WindowState[= State%]
```

For example:

```
CrystalReport1.WindowState= 2
```

«When the report is printed to a preview window, the preview window appears maximized when opened.»

Remarks

Use the following values to set the WindowState property:

- **0 = Normal**

The preview window appears neither minimized nor maximized. It appears in a default size and position previously defined by your application or by Windows.

- **1 = Minimized**

The preview window appears minimized as an icon close to the lower left hand corner of the screen. The icon can be restored to display the window in a normal state.

- **2 = Maximized**

The preview window is maximized when opened to fill the entire screen.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowTitle

Description

Specifies the title you want to appear in the preview window title bar when the report is printed to a window.

Usage

```
[form.]Report.WindowTitle[= Title$]
```

For example:

```
CrystalReport1.WindowTitle = "Quarterly Earnings"
```

«Sets the title of the preview window (the string that appears on the title bar) to “Quarterly Earnings”.»

Remarks

- Make sure that the title is enclosed in quotes.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, = 0.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowTop

Description

Sets the distance, in pixels, that the preview window is to appear from the top edge of the parent window. If the preview window is a top level window, then the distance is measured from the top edge of the screen.

Usage

```
[form.]Report.WindowTop [= Distance%]
```

For example:

```
CrystalReport1.WindowTop = 100
```

«Sets the top edge of the preview window 100 pixels from the top of the screen.»

Remarks

- If you are not satisfied with the default setting, enter the number of pixels you want between the top of the screen and the top of your window.

- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

WindowWidth

Description

Specifies the width of the preview window in pixels.

Usage

```
[form.]Report.WindowWidth [= Width%]
```

For example:

```
CrystalReport1.WindowWidth = 480
```

«Specifies a preview window 480 pixels wide.»

Remarks

- If you are not satisfied with the default setting, enter the external width of your window, in pixels.
- Select a value here only if you are printing to a window, if *Destination, Page 502*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow, Page 257

METHODS

FetchSelectionFormula

Description

FetchSelectionFormula returns the selection formula from the current report.

Usage

```
[form.]Report.FetchSelectionFormula
```

For example:

```
SelectionFormula$= CrystalReport1.FetchSelectionFormula
```

«Retrieves the selection formula from CrystalReport1.»

Remarks

This method does not populate *SelectionFormula, Page 566*, and it does not conflict with setting the property. Both the method and the property can be used in the same code.

Availability

Runtime only

GetNSubreports

Looks at the report specified in *ReportFileName, Page 557*, and returns the number of subreports in that report.

Usage

```
[form.]Report.GetNSubreports
```

For example:

```
Number=CrystalReport1.GetNSubreports  
«Returns the number of subreports in CrystalReport1.»
```

Remarks

Zero indexed. Thus, if the function returns 2, that indicates that there are three subreports, the first one being subreport 0.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

GetNthSubreportName

Looks at the report specified in the ReportFileName property and returns a string which is the name of the nth subreport in that report.

Usage

```
[form.]Report.GetNthSubreportName (SubreportNum%)
```

For example:

```
SubreportName=CrystalReport1.GetNthSubreportName (2)  
«Returns the name of the third subreport in CrystalReport1.»
```

Remarks

The valid range for the parameter is 0 to n-1, where n is the number you get back from GetNSubreports.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

LogoffServer

Terminates the specified database connection established earlier with the *LogonServer*, [Page 596](#).

Usage

```
[form.]Report.LogoffServer (connectionId%, boolean allConnections)
```

For example:

```
CrystalReport1.LogoffServer (1, False)  
«Terminates database connection 1 and only that connection.»
```

Parameters

Parameter	Description
connectionId	Integer value that specifies a specific database connection established earlier with <i>LogonServer</i> , <i>Page 596</i> . If you have set allConnections to True, connectionId should be set to 0.
allConnections	Boolean value that specifies whether or not to terminate ALL database connections that have been established with the LogonServer method. <ul style="list-style-type: none">● True = Terminate all connections.● False = Terminate only the specified connection.

LogonServer

Logs on to the specified server and returns a unique connection id which can be used to log off of this server using the *LogoffServer*, *Page 595*.

Usage

```
[form.]Report.LogonServer (dllName$, ServerName$, DatabaseName$,  
UserID$, Password$)
```

For example:

```
connectionId% = CrystalReport1.LogonServer ("pdsodbc.dll",  
"Accounting", "Administration", "bobg", "bigboard")
```

«Connects to the “Administration” database via the “Accounting” data source using the user ID “bobg” and the password “bigboard”.»

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log onto, for example, "PDSODBC.DLL". Note that the dllName must be enclosed in quotes. DLL names have the following naming convention: <ul style="list-style-type: none">● PDB*.DLL for standard (non-SQL) databases.● PDS*.DLL for SQL/ODBC databases.
ServerName	Specifies the logon name for the server used to create the report. *For ODBC, use the data source name.
DatabaseName	Specifies the logon name for the database used to create the report.
UserID	Specifies the user ID necessary to log on to the server.
Password	Specifies the password necessary to log on to the server. When you are using this structure to retrieve information using <i>PEGetNthTableLogOnInfo</i> , <i>Page 205</i> , the password parameter is undefined.

PageCount

Returns the number of pages in the report.

Usage

```
[form.]Report.PageCount
```

For example:

```
NumberofPages = CrystalReport1.PageCount
```

«Returns 100 if the number of pages in the specified report is 100.»

Remarks

PageCount must be called after the *Action*, *Page 740*, to get a valid page count.

PageFirst

Displays the first page of the report in the preview window.

Usage

```
[form.]Report.PageFirst
```

For example:

```
CrystalReport1.PageFirst
```

«Displays the first page of the report to the preview window.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PageLast

Displays the last page of the report in the preview window.

Usage

```
[form.]Report.PageLast
```

For example:

```
CrystalReport1.PageLast
```

«Displays the last page of the report in the preview window.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PageNext

Displays the next page of the report in the preview window.

Usage

```
[form.]Report.PageNext
```

For example:

```
CrystalReport1.PageNext
```

«Displays the next page of the report in the preview window.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PagePrevious

Displays the previous page of the report in the preview window.

Usage

```
[form.]Report.PagePrevious
```

For example:

```
CrystalReport1.PagePrevious
```

«Displays the previous page of your report in the preview window.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PageShow

Displays a specific page of the report in the preview window.

Usage

```
[form.]Report.PageShow [(PageToShow%)]
```

For example:

```
CrystalReport1.PageShow (3)
```

«Shows the third page of the report.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PageZoom

Description

Sets the magnification factor for the report in the preview window to a value from 25% to 400% of the actual size.

Usage

```
[form.]Report.PageZoom [(PercentZoomLevel%)]
```

For example:

```
CrystalReport1.PageZoom (150)
```

«Sets the magnification/zoom level to 150% for the current.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.

PageZoomNext

Description

Zooms the magnification level of the report in a preview window to the next default zoom level.

Usage

```
[form.]Report.PageZoomNext
```

For example:

```
CrystalReport1.PageZoomNext  
«Sets the magnification/zoom level to the next default level.»
```

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action, Page 740*.
- Default zoom levels are Full Page, Fit One Side, and Fit Both Sides. These levels correspond to the levels available by clicking the Zoom button on the Seagate Crystal Reports toolbar.

PrinterSelect

Description

Displays the Printer Selection common dialog box which enables the user to specify a different printer.

Usage

```
[form.]Report.PrinterSelect
```

For example:

```
CrystalReport1.PrinterSelect  
«When the user prints, a dialog box appears for selecting the desired printer for printing the report.»
```

Remarks

- This method allows you to pick the printer used for printing the report.
- This method is intended primarily for printing reports to a printer. However, changes made to the Print Setup dialog box may also affect how reports appear in preview windows and when exported.

PrintReport

Description

PrintReport triggers the printing of the report.

Usage

```
[form.]Report.PrintReport
```

For example:

```
Result% = CrystalReport1.PrintReport
```

«Prints the specified report.»

Remarks

- PrintReport returns a result code, 0 if the call is successful, an error code in the 20XXX range if it fails.
- You can also print a report using the *Action*, *Page 740*. If something goes wrong, however, you get a runtime error that will terminate your application. For this reason, you will need to set up an error handler.

NOTE: If you are currently using the OCX/VBX control in your application, you will not be able to print individual subreports.

Availability

Runtime

Related Report Engine Functions

PEStartPrintJob, *Page 348*

ReplaceSelectionFormula

Description

ReplaceSelectionFormula overrides the selection formula from the current report with the string that is passed.

Usage

```
[form.]Report.ReplaceSelectionFormula [(SelectionFormulaString$)]
```

For example:

```
CrystalReport1.ReplaceSelectionFormula ("{Company.State}='CA'")
```

«Uses “{Company.State}='CA'” as the selection formula for the report.»

Remarks

This method DOES NOT use the string in *SelectionFormula*, *Page 566*, and DOES conflict with setting the property. You can not set the SelectionFormula property and call ReplaceSelectionFormula in the same code sequence. A Visual Basic error condition will be raised in such a case.

Availability

Runtime only

Reset

Resets the value of all properties (except the *DataSource*, *Page 502*) to their default values.

Usage

```
[form.]Report.Reset
```

For example:

```
CrystalReport1.Reset
```

«Returns all properties (except DataSource) for CrystalReport1 to their default values.»

RetrieveDataFiles

Description

RetrieveDatafiles retrieves all “table” locations from the current report, populates *DataFiles*, *Page 500*, and returns the number of “tables” in the report.

Usage

```
[form.]Report.RetrieveDatafiles
```

For example:

```
NumberofDatafiles% = CrystalReport1.RetrieveDatafiles
```

«Populates the DataFiles property with the table locations from CrystalReport1.»

Remarks

This method can only be called AFTER *ReportFileName*, *Page 557*, has been set.

Availability

Runtime only

RetrieveLogonInfo

Description

RetrieveLogonInfo retrieves logon information (except for the password) for all “tables” in the current report, populates *LogOnInfo*, *Page 526*, and returns the number of “tables” in the report.

Usage

```
[form.]Report.RetrieveLogonInfo
```

For example:

```
NumberOfTables% = CrystalReport1.RetrieveLogonInfo  
«Retrieves the logon information for all the tables in CrystalReport1.»
```

Remarks

This method can only be called AFTER the *ReportFileName*, *Page 557*, has been set. This method DOES NOT use the string in the Connect property and DOES conflict with setting the property. You can not set the Connect property and call RetrieveLogonInfo in the same code sequence. A Visual Basic error condition will be raised in this case.

Availability

Runtime only

RetrieveSQLQuery

Description

RetrieveSQLQuery retrieves the SQL Query from the current report and populates .

Usage

```
[form.]Report.RetrieveSQLQuery
```

For example:

```
CrystalReport1.RetrieveSQLQuery  
«Retrieves the SQL query from CrystalReport1.»
```

Remarks

This method can only be called AFTER *ReportFileName*, *Page 557*, has been set.

Availability

Runtime only

RetrieveStoredProcParams

Description

RetrieveStoredProcParams retrieves all stored procedure parameters from the current report, populates *StoredProcParam*, *Page 571*, and returns the number of parameters.

Usage

```
[form.]Report.RetrieveStoredProcParams
```

For example:

```
NumberofParams% = CrystalReport1.RetrieveStoredProcParams  
«Retrieves the stored procedure parameters from CrystalReport1.»
```

Remarks

This method can only be called AFTER *ReportFileName*, *Page 557*, has been set.

Availability

Runtime only

SpecifyDataSourceField

Enables you to specify the columns that appear, their order, and their width for reports that are automatically generated from a Data control. If you call this function one or more times, only the columns indicated by the calls will appear in the report. You must call this function one time for each column you are setting.

Usage

```
[form.]Report.SpecifyDataSourceField (ColumnNum%, ColumnName$,  
ColumnWidth%)
```

For example:

```
CrystalReport1.SpecifyDataSourceField (0, "Year Born", 10)
```

```
CrystalReport1.SpecifyDataSourceField (1, "Au_ID", 20)
```

«If the data control was pointing to the Authors table in the Biblio sample, the code results in a report where the first column is year born and the second column is author_id.»

Remarks

- This method only works when ReportSource = 3, Data Control Fields. It does not work when ReportSource = 1, Bound TrueDBGrid Control.
- The first parameter, ColumnNum, specifies the column number (zero indexed).
- The second parameter, ColumnName, is the name of the column as it appears in the data control.
- The third parameter is the column width, in characters.
- To use the default width for the type of column, pass -1 as the third parameter.

Error Messages

Error #	Error	Message/Explanation
400	PE_ERR_ CONFLICTLOGONINFOCONNECT	<p>“Do not use both Connect property and LogonInfo property.”</p> <p>This message appears if you try to set both of these properties at the same time.</p>
401	PE_ERR_ REPORTFILENAMENOTSET	<p>“ReportFileName property must be set before calling method.”</p> <p>Some methods retrieve information from a report. This message appears if you try to call these methods before setting the ReportFileName property. The methods are:</p> <ul style="list-style-type: none">● FetchSelectionFormula● ReplaceSelectionFormula● RetrieveDataFiles● RetrieveSQLQuery● RetrieveStoredProcParams● GetNSubreports
402	PE_ERR_ SELECTIONFORMULACONFLICT	<p>“Do not use both SelectionFormula and ReplaceSelectionFormula.”</p> <p>ReplaceSelectionFormula is a method. This message appears if you attempt to use it after SelectionFormula.</p>
403	PE_ERR_ BADDLL	<p>“DLL name must be specified.”</p> <p>This message appears if you call the LogOnServer method and don't specify the DLLname parameter.</p>
404	PE_ERR_ STRINGTOOLONG	<p>“A string is too long.”</p> <p>This message appears:</p> <ul style="list-style-type: none">● if you set an element of the ParameterFields property to a string whose value part is greater than 256 characters, or● if you call the LogOnServer method, and the server, database, userid, or password parameters are greater than 128 characters.

<i>Error #</i>	<i>Error</i>	<i>Message/Explanation</i>
405	PE_ERR_ BADSUBREPORTINDEX	“Invalid Subreport index.” This message appears if you call the method GetNthSubreportName and the first parameter does not correspond to a subreport.
406	PE_ERR_ BADSUBREPORTNAME	“Unknown Subreport name.” This message appears if you set the SubreportToChange property to the name of a subreport that does not exist in the current report.
407	PE_ERR_ ODBCTABLENOTSET	“Need ODBC table name for export.” This message appears when you attempt to export to ODBC without setting the PrintFileODBCTable property.

9

Crystal Report Engine Object Library

What you will find in this chapter...

Object Hierarchy, Page 610

Object Naming Conflicts, Page 611

Object Library Events, Page 611

Crystal Report Engine Object Library Reference, Page 611

Object Hierarchy

The Crystal Report Engine Object Library for the Crystal Report Engine Automation Server provides all of the power of the Crystal Report Engine in an easy to use object-oriented model. The following diagram illustrates this object hierarchy within the Object Library:



Object Naming Conflicts

If your project includes other libraries that contain objects named identical to those found in the Crystal Report Engine Object Library, you will encounter conflicts unless you reference the objects using a prefix. For example, if you have included the DAO Library with the Crystal Report Engine Object Library in your project, both libraries include a Database Object. In order to avoid conflicts, you must prefix the objects as follows:

CRPEAuto.Database

for the Crystal Report Engine Object Library, or

DAO.Database

for the DAO Library.

NOTE: You must always prefix when referencing the Font Object. Visual Basic includes a VB Font Object that is always included when creating a project, so conflicts will always occur unless the object is properly referenced. For example, CRPEAuto.Font.

Object Library Events

Events are only available when using Visual Basic 5.0. To use events you must first declare the desired Object (Report or Window) as Public and With Events in the General, Declarations area of the module.

For example, declare the Report object as follows:

```
Public WithEvents reportVar As Report
```

The Visual Basic code screen will then list reportVar in the Object list, with events listed in the Procedure list. Adding code to an event will cause this code to be run whenever the event occurs, which in turn will be followed by the default behavior for the event. For complete information on handling events, see *Handling Preview Window Events, Page 89*.

NOTE: Event variables can only be set once. Use a new variable for each Report/Window Object declared with events.

Crystal Report Engine Object Library Reference

Application Object

The **Application** Object is the only creatable object in the Crystal Report Engine Object Library. All access to the Crystal Report Engine Automation Server must begin with the creation of an instance of the Application object.

An instance of the Application object can be created using the Visual Basic CreateObject function and the Prog Id Crystal.CRPE.Application. For example:

```
Dim app As Application  
Set app = CreateObject("Crystal.CRPE.Application")
```

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
LastErrorCode	Returns the most recent error code. Used for errors at the application level	Read only
LastErrorString	Returns the most recent error string. Used for errors at the application level.	Read only
Options	Returns the <i>GlobalOptions Object</i> , <i>Page 650</i> , which allows you to request more descriptive error messages from the Crystal Report Engine.	Read only
Parent	Returns a null object (Nothing) as Application has no parent.	Read only

NOTE: When any Visual Basic standard error is returned, LastErrorString will be empty.

Methods

CanClose

The **CanClose** method indicates whether or not the *Application Object*, *Page 611*, can be destroyed. This method will return FALSE as long as there are valid Report objects in existence. The Application object can only be destroyed if no instances of the *Report Object*, *Page 685*, exist.

Syntax

```
object.CanClose
```

Returns

- **TRUE** - if the Engine can be closed.
- **FALSE** - if the Engine is busy.

ClearError

The **ClearError** method clears the application error code and string stored when calling the LastErrorCode and LastErrorString properties.

Syntax

```
object.ClearError
```

LogOffServer

The **LogOffServer** method logs off an SQL server or ODBC data source. Use this method when you have logged on to the data source using *LogOnServer*, *Page 614*.

Syntax

```
object.LogOffServer DLLName, ServerName, _DatabaseName,  
UserID, Password
```

Parameters

Parameter	Type
DLLName	Specifies the name of the DLL for the server or password protected non-SQL table you want to log on to, for example “PDSODBC.DLL”. Note that the DLLName must be enclosed in quotes. DLL names have the following naming convention: PDB*.DLL for standard (non-SQL) databases, PDS*.DLL for SQL/ODBC databases.
ServerName	Specifies the log on name for the server used to create the report.* (For ODBC, use the data source name.) This value is case-sensitive.
DatabaseName (Optional)	Specifies the name for the database used to create the report.*
UserID (Optional)	Specifies the User ID number necessary to log on to the server.*
Password (Optional)	Specifies the password necessary to log on to the server.

*When you pass an empty string (“”) for this parameter, the program uses the value that's already set in the report. If you want to override a value that's already set in the report, use a non-empty string (i.e., “Server A”).

Remarks

If you try to log off a server that is still in use (i.e., there is an object variable still in focus that holds reference to a report that requires being logged on to the server to access data) you will be unable to log off. This will apply to every object that comes from the *Report Object*, *Page 685*, as they all hold reference to the report through their respective Report properties.

LogOnServer

The **LogOnServer** method logs on to an SQL server or ODBC data source. Once logged on using this method, you will remain logged on until you call *LogOffServer*, *Page 613*, or until the *Application Object*, *Page 611*, is destroyed. This method corresponds to *PLogOnServer*, *Page 241*, of the Crystal Report Engine API.

Syntax

```
object.LogOnServer DLLName, ServerName, _DatabaseName,  
UserID, Password
```

Parameters

Parameter	Type
DLLName	Specifies the name of the DLL for the server or password protected non-SQL table you want to log on to, for example “PDSODBC.DLL”. Note that the dllName must be enclosed in quotes. DLL names have the following naming convention: PDB*.DLL for standard (non-SQL) databases, PDS*.DLL for SQL/ODBC databases.
ServerName	Specifies the log on name for the server used to create the report.* (For ODBC, use the data source name.) This value is case-sensitive.
DatabaseName (Optional)	Specifies the name for the database used to create the report.*
UserID (Optional)	Specifies the User ID number necessary to log on to the server.*
Password (Optional)	Specifies the password necessary to log on to the server.

*When you pass an empty string (“”) for this parameter, the program uses the value that's already set in the report. If you want to override a value that's already set in the report, use a non-empty string (i.e., “Server A”).

OpenReport

The **OpenReport** method opens an existing report file, creating an instance of the Report object. Through the *Report Object*, *Page 685*, you can change formatting, formulas, selection formulas, and sort fields for the report, then print, preview, or export the report. This method corresponds to *POpenPrintJob*, *Page 248*, of the Crystal Report Engine API.

Syntax

```
Dim rep As Report  
Set rep = app.OpenReport(ReportFilePath)
```

Parameters

Parameter	Description
ReportFilePath	Specifies a string value indicating the file name and path of the report you want to open.

Returns

Returns an instance of the *Report Object*, *Page 685*, if the report was successfully opened. Returns 0 if the report file does not exist or if an error occurs.

Area Object

The **Area** Object represents an area in a report. An area is a group of like sections in the report (i.e., Details A - Da, Details B - Db, etc.) that all share the same characteristics. Each section within the area can be formatted differently. This object allows you to retrieve information and set options for a specified area in your report.

Properties

Property	Description	Read/Write
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
CopiesToPrint	Returns/Sets the number of copies of each item in the Details section of the report. For example, by default, each line of the Details section only prints once. By setting this to 3, each line of the Details section would print 3 times.	Read/Write
GroupNumber	If the area is a group, this returns the group number. Otherwise, exception is thrown.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
GroupOptions	Returns <i>GroupAreaOptions Object</i> , Page 656 , which provides properties/methods for getting and setting group options. Exception is thrown if area is not a group area.	Read only
Kind	Returns CRAreaKind (see table below) which specifies what “kind” of area (i.e., Details, Report Header, Page Footer, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crDetail	4
crGroupFooter	5
crGroupHeader	3
crPageFooter	7
crPageHeader	2
crReportFooter	8
crReportHeader	1

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Options	Returns <i>AreaOptions Object</i> , Page 617 , which provides properties for getting and setting area options (i.e., new page before, keep together, etc.).	Read only
Parent	Reference to the Parent object (<i>Area Object</i> , Page 615).	Read only
Report	Reference to <i>Report Object</i> , Page 685 .	Read only
Sections	Returns <i>Sections Collection</i> , Page 705 , which specifies a collection of all the sections in the area.	Read only

AreaOptions Object

The **AreaOptions** object allows you to retrieve information and set options for a specified area of your report (i.e., new page before, keep together, underlay section, etc.). An AreaOptions Object is obtained from the Options property of the *Area Object, Page 615*.

Each report area is comprised of sections which can be formatted independently of each other using the *SectionOptions Object, Page 701*. These section options are combined with the options set using the AreaOptions Object to make up the final appearance of a section and area of the report. The settings in the AreaOptions object will affect all sections within the area, while settings in the SectionOptions object will affect only the section they are set to.

If there is a conflict of settings between the AreaOptions object and the SectionOptions object, the object with an option set to TRUE will override the setting for the other object. For example, if the AreaOptions object has the KeepTogether property set to TRUE, all sections within the area will have KeepTogether applied, even if the SectionOptions object for a section has KeepTogether set to FALSE. If, however, the AreaOptions object has KeepTogether set to FALSE, but a section within that area has KeepTogether set to TRUE, that section will have the KeepTogether format option applied.

If an area has only a single section, all options will be combined between both the AreaOption object and the SectionOptions object. All TRUE settings set in either object will result in a TRUE setting for the entire area and section. While changing format options for areas and sections in reports, be sure to keep track of settings in both the AreaOptions and SectionOptions objects.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
KeepTogether	Returns/Sets Boolean value indicating if the values in this area are kept together across pages.	Read/Write
KeepTogetherFormula	Returns/Sets string specifying a formula for conditionally setting when values in the area are kept together across pages.	Read/Write
NewPageAfter	Returns/Sets Boolean value indicating if a page break occurs just after this area.	Read/Write

Property	Description	Read/Write
NewPageAfterFormula	Returns/Sets string specifying a formula for conditionally setting when a page break occurs after the area.	Read/Write
NewPageBefore	Returns/Sets Boolean value indicating if a page break occurs just before this area.	Read/Write
NewPageBeforeFormula	Returns/Sets string specifying a formula for conditionally setting when a page break occurs before the area.	Read/Write
NotHideForDrillDown	Returns/Sets Boolean value indicating if the area is not hidden for drill-down.	Read/Write
NotHideForDrillDown-Formula	Returns/Sets string specifying a formula for conditionally setting when the area is not hidden for drill-down.	Read/Write
Parent	Reference to the Parent object (<i>Area Object, Page 615</i>).	Read only
PrintAtBottomOfPage	Returns/Sets Boolean value indicating if the area is to appear at the bottom of the page.	Read/Write
PrintAtBottomOfPage-Formula	Returns/Sets string specifying a formula for conditionally setting when the area is printed at the bottom of the page.	Read/Write
Report	Reference to <i>Report Object, Page 685</i> .	Read only
ResetPageNumberAfter	Returns/Sets Boolean value indicating if the page number is reset to one after this area is printed.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ResetPageNumberAfter-Formula	Returns/Sets string specifying a formula for conditionally setting when the page number should be reset after the area is printed.	Read/Write
Visible	Returns/Sets Boolean value indicating whether the report area is visible or not.	Read/Write
VisibleFormula	Returns/Sets string specifying a formula for conditionally setting when an area is visible.	Read/Write

Areas Collection

The **Areas** Collection contains a collection of area objects for every area in the report. Access a specific *Area Object*, *Page 615*, in the collection using the *Item* property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Count	Returns the number of areas in the collection.	Read only
Item	Returns <i>Area Object</i> , <i>Page 615</i> . Item has an index parameter that can be either a string reference to the area (i.e., “RH”, “PH”, “GHn”, “D”, “GFn”, “PF”, or “RF”) or a numeric, 1-based index (i.e., Item (1) for the Report Header area. The items in the collection are indexed in the order they are listed for each section/area.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference an area directly; for example, Areas ("RH") or Areas (1).

BlobField Object

The **BlobFieldObject** Object allows you to get and set information for bitmap database fields in a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Field	Returns <i>DatabaseFieldDefinition Object, Page 624</i> , containing information about the BLOB field.	Read only
Kind	Returns CRObjectKind object which specifies what "kind" of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3

<i>Constant</i>	<i>Value</i>
crOLEObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Box Object

The **Box Object** represents a box that has been drawn on the report. This object allows you to get information about boxes in a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjectKind object which specifies what “kind” of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOLEObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

CrossTabObject Object

The **CrossTabObject** Object allows you to get and set information for cross-tab objects in a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjectKind object which specifies what “kind” of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOleObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
SummaryFields	Returns the <i>SummaryFieldDefinitions Collection, Page 715</i> , providing access to information about all summary fields in the cross-tab object.	Read only

Database Object

The **Database Object** provides properties to get information about the database accessed by a report. See *Object Naming Conflicts, Page 611*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Parameters	Returns <i>DatabaseParameters Collection, Page 629</i> , which specifies the database parameters used in the report.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
Tables	Returns <i>DatabaseTables Collection, Page 635</i> , which specifies the database objects used in the report (i.e., an Access report may contain a query or SQL Server report may be based on a stored procedure - if so, they will be returned as part of this collection along with the database table used in the report).	Read only

Methods

Verify

The **Verify** method verifies that the location of the database is still valid and checks to see if any changes have been made to table design, etc. If there are any changes to the database, the Verify method will update the report automatically to reflect these changes.

Syntax

```
object.Verify
```

DatabaseFieldDefinition Object

The **DatabaseFieldDefinition** Object represents a database field used in the report. This object provides properties for getting information on database fields in the report.

Properties

<i>Property</i>	<i>Definition</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
DatabaseFieldName	Specifies the name of the field in the database (i.e., Product ID).	Read only
Kind	Returns CRFieldKind which specifies what “kind” of field (i.e., database, summary, formula, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crDatabaseField	1
crFormulaField	2
crGroupNameField	5
crParameterField	6
crSpecialVarField	4
crSummaryField	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the field within the report (table.FIELD). For example, product.PRODUCT ID.	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
TableAliasName	Specifies the alias name used in the report to reference the database table. By default this is the name of the database table.	Read only
ValueType	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7

<i>Constant</i>	<i>Value</i>
crOLEField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

DatabaseFieldDefinitions Collection

The **DatabaseFieldDefinitions** Collection is a collection of database field definition objects. One object exists in the collection for every database field accessed by the report. Access a specific *DatabaseFieldDefinition Object*, *Page 624*, in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Count	The number of <i>DatabaseFieldDefinition Object</i> , <i>Page 624</i> , in the collection.	Read only
Parent	Reference to the Parent object (<i>DatabaseTable Object</i> , <i>Page 630</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Item	Returns <i>DatabaseFieldDefinition Object</i> , <i>Page 624</i> . Item has an index parameter that can be either a string reference to the database field name or a numeric, 1-based index (i.e., Item (1)) for the first database field in the collection. When a numeric index is used, the items in the collection are indexed in the order they were added to the report.	Read only

Remarks

Instead of using the Item property as shown, you can reference a database directly, for example, DatabaseFieldDefinition("Product ID") or DatabaseFieldDefintion(1).

DatabaseParameter Object

The **DatabaseParameter** Object specifies a stored procedure in a SQL database or an Access parameterized query.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Name	Returns the name of the parameter from the parameterized query/stored procedure that the report was created from.	Read only
Parent	Reference to the Parent object (<i>Database Object</i> , <i>Page 623</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Type	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7
crOleField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Value	Returns/Sets the value for a database parameter as a string (i.e., when you are setting a numeric parameter you would use: DatabaseParameter.Value = “5”).	Read/Write

DatabaseParameters Collection

The **DatabaseParameters** Collection is a collection of database parameter objects. A DatabaseParameter object exists in the collection for every stored procedure parameter accessed by the report. Access a specific *DatabaseParameter Object*, *Page 627*, in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Count	Specifies the number of database parameters in the collection.	Read only
Item	Returns <i>DatabaseParameter Object</i> , <i>Page 627</i> . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)). The items in the collection are indexed in the order they were added to the report.	Read only
Parent	Reference to the Parent object (<i>Database Object</i> , <i>Page 623</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a parameter directly, for example, DatabaseParameters(1).

DatabaseTable Object

The **DatabaseTable** Object refers to a database table accessed by the report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
DescriptiveName	Returns a descriptive name for the data source (i.e., Access table returns "Microsoft DAO Database DLL").	Read only
DllName	Returns the name of the DLL used in the report (i.e., PDBDAO.DLL, PDSODBC.DLL, etc.).	Read only
Fields	Returns <i>DatabaseFieldDefinitions Collection</i> , <i>Page 626</i> , specifying a collection of database fields in the table.	Read only
Location	Gets/sets the location of the database table.	Read/Write
LogOnDatabaseName	Returns the name of the database if the database was logged on to. The format of this name will depend on the data source.	Read only
LogOnServerName	Returns the name of the server that the database logged on to (i.e., with ODBC reports, the ODBC data source name is returned).	Read only
LogOnUserID	Returns user ID used when logging on to the data source. ·	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns/Sets the alias name for the database table used in the report.	Read/Write
Parent	Reference to the Parent object (<i>Database Object</i> , Page 623).	Read only
PrivateDataTag	This property contains information about the type of data accessed by the database driver. This value is used by the database driver for the data source and should not be used in your application. If your report has been designed using the <i>Crystal Active Data Driver</i> , Page 93, the PrivateDataTag property will contain a value of 3 once an active data source has been assigned to the report. This value is associated with <i>GetPrivateData</i> , Page 632, and <i>SetPrivateData</i> , Page 633, methods.	Read only
Report	Reference to <i>Report Object</i> , Page 685.	Read only
SessionUserID	Returns the user ID used for Access session security. Returns the user ID used to create the report, or the user ID passed to the <i>SetSessionInfo</i> , Page 633, method if an Access session has been opened.	Read only
Type	Returns value of type CRDatabaseType (see table below) indicating the type of database (i.e., standard or SQL).	Read only

<i>Constant</i>	<i>Value</i>
crSQLDatabase	2

<i>Constant</i>	<i>Value</i>
crStandardDatabase	1

Methods

GetPrivateData

The **GetPrivateData** method is used to retrieve the current private data for a given table. The private data will be returned as the type specified in the VariantType parameter. The format of the private data is specific to each database driver.

For example, if the report has been designed using the *Crystal Active Data Driver, Page 93*Crystal . **GetPrivateData** can be used to obtain the active data object assigned to the report using *SetPrivateData, Page 633*.

Syntax

```
object.GetPrivateData VariantType
```

Parameters

<i>Parameter</i>	<i>Description</i>
VariantType	A constant value indicating the type of data being retrieved. Use a Visual Basic VarType constant value. For example, vbDataObject will return an Active Data Object such as a DAO Recordset.

Returns

Variant.

SetLogOnInfo

The **SetLogOnInfo** method logs on to the data source so table data can be accessed.

Syntax

```
object.SetLogOnInfo DllName, ServerName, _
    DatabaseName, UserID, Password
```

Parameters

Parameter	Description
DllName	Specifies the DLL used to access the database by the Crystal Report Engine (i.e., PDSODBC.DLL).
ServerName	Specifies the name of the server or ODBC data source where the database is located (i.e., CRSS).
DatabaseName (Optional)	Specifies the name of the database.
UserID (Optional)	Specifies a valid user name for logging on to the data source.
Password (Optional)	Specifies a valid password for logging on to the data source.

SetPrivateData

The **SetPrivateData** method is used to provide information about a data source to the database driver associated with this DatabaseTable object. For instance, if a report has been designed using the *Crystal Active Data Driver, Page 93*, this method can be used to provide an active data source for the report, such as a DAO, ADO, or RDO Recordset or a CDO Rowset. In this case, the object passed to the second parameter of this method replaces, at runtime, the field definition file used to create the report. For complete information, see *Using the Active Data Driver, Page 93*.

Syntax

```
object.SetPrivateData DataTag, Data
```

Parameters

Parameter	Description
DataTag	A value indicating the type of data being passed to the DatabaseTable object in the Data parameter. Currently, the only possible value is 3. This value must be used for all Active data sources including DAO, ADO, RDO, CDO, and the Visual Basic data control.
Data	Variant data passed to the database driver. For example, with Active data, this must be a Recordset object if you are using DAO, ADO, or the Visual Basic data control. This must be a Rowset object if you are using CDO.

SetSessionInfo

The **SetSessionInfo** method allows the user to log on to a secured Access session.

Syntax

```
object.SetSessionInfo SessionUserID As String, _  
SessionPassword As String
```

Parameters

Parameter	Description
SessionUserID	Specifies the Access userID used to log on to an Access session.
SessionPassword	Specifies the session password for Access secured session.

Remarks

In Microsoft Access 95 and later, an Access database can have session security (also known as user-level security), database-level security, or both. If the Access database contains only session security, simply pass the session password to the SessionPassword parameter. If the Access database contains database-level security, use a linefeed character, Chr(10), followed by the database-level password. For example:

```
object.SetSessionInfo "userID", Chr(10) & "dbpassword"
```

If the Access database contains both session security and database-level security, use the session password followed by the linefeed character and the database password:

```
object.SetSessionInfo "userID", "sesspswd" & _  
Chr(10) & "dbpassword"
```

Alternately, database-level security can also be handled by assigning the database-level password to the Password parameter of the *SetLogOnInfo*, *Page 632*, method.

TestConnectivity

The **TestConnectivity** method tests to see if the database can be logged on to with the current information and if the database table can be accessed by the report.

Syntax

```
object.TestConnectivity
```

Returns

TRUE (1) if the database session, log on, and location information is all correct; FALSE (0) if something is wrong.

DatabaseTables Collection

The **DatabaseTables** Collection is a collection of DatabaseTable objects. A DatabaseTable object exists for every database object (i.e., table, query, stored procedure, etc.) accessed by the report. Access a specific *DatabaseTable Object*, *Page 630*, in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Count	Returns the number of database objects in the collection.	Read only
Item	Returns <i>DatabaseTable Object</i> , <i>Page 630</i> . Item has an index parameter that can be either a string reference to the table, (i.e., Item("Categories")) or a numeric, 1-based index (i.e., Item (1)). When a numeric index is used, the items in the collection are indexed in the order they were added to the report.	Read only
Parent	Reference to the Parent object (<i>Database Object</i> , <i>Page 623</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a table directly, for example, DatabaseTable("Categories") or DatabaseTable(1).

EventInfo Object (32-bit only)

Properties

Property	Description	Read/Write
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
ActivatePrintWindow-EventEnabled	Returns/Sets a Boolean value indicating whether the <i>ActivatePrintWindow</i> , <i>Page 725</i> , or <i>DeactivatePrintWindow</i> , <i>Page 727</i> , events of the <i>Window Object</i> , <i>Page 724</i> , are enabled.	Read/Write
ClosePrintWindowEvent-Enabled	Returns/Sets a Boolean value indicating whether the <i>ClosePrintWindow</i> , <i>Page 726</i> , event of the <i>Window Object</i> , <i>Page 724</i> , is enabled.	Read/Write
GroupEventEnabled	Returns Boolean value indicating whether the <i>GroupTreeButtonClicked</i> , <i>Page 730</i> , <i>ShowGroup</i> , <i>Page 734</i> , or <i>DrillOnGroup</i> , <i>Page 728</i> , events of the <i>Window Object</i> , <i>Page 724</i> , are enabled.	Read only
Parent	Reference to the Parent object (<i>Report Object</i> , <i>Page 685</i>).	Read only
PrintWindowButton-EventEnabled	Returns Boolean value indicating if events corresponding to preview window buttons for the <i>Window Object</i> , <i>Page 724</i> , are enabled.	Read only
ReadingRecordsEvent-Enabled	Returns Boolean value indicating if the <i>ReadingRecords</i> , <i>Page 694</i> , event of the <i>Report Object</i> , <i>Page 685</i> , is enabled.	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
StartStopEventEnabled	Returns Boolean value indicating if the <i>Start</i> , <i>Page 694</i> , and <i>Stop</i> , <i>Page 695</i> , events of the <i>Report Object</i> , <i>Page 685</i> , are enabled.	Read only

ExportOptions Object

The **ExportOptions** object provides properties and methods for retrieving information and setting options for exporting your report (i.e., export format, destination, etc.). An ExportOptions Object is obtained from the ExportOptions property of the *Report Object*, *Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
CharFieldDelimiter	Returns/Sets the character used to separate fields in character separated text formats. This character delimits every field in the file.	Read/Write
CharStringDelimiter	Returns/Sets the character used to separate strings in character separated text formats. This character delimits only string fields (numeric, date fields, etc., have no delimiter).	Read/Write
DestinationDLLName	Returns the name of the DLL used to export the report to a specific destination.	Read only
DestinationType	Returns/Sets CRExportDestinationType (see table below) indicating the destination type for the exported report (i.e., disk, mail, etc.).	Read/Write

<i>Constant</i>	<i>Value</i>
crEDTDiskFile	1
crEDTMailMAPI	2
crEDTMailVIM	3
crEDTMicrosoftExchange	4
crEDTNoDestination	0

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
DiskFileName	Returns/Sets the file name if the report is exported to a disk.	Read/Write
ExcelTabHasColumn-Headings	Returns/Sets Boolean value indicating whether when exporting to Excel format, the spreadsheet will be displayed including column headings. By default this property is set to FALSE.	Read/Write
ExchangeDestination-Type	Returns/Sets CRExchangeDestinationType (see table below) indicating the Exchange destination type for reports exported to Exchange folders.	Read/Write

<i>Constant</i>	<i>Value</i>
crExchangePostDocMessage	1011
crExchangeFolderType	0

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ExchangeFolderPath	Returns/Sets the path of the Exchange folder for reports exported to Exchange (i.e., "PersonalFolders@Inbox").	Read/Write
ExchangePassword	Returns/Sets the password used to access the Exchange folder for reports exported to the 16-bit version of Exchange.	Read/Write
ExchangeProfile	Returns/Sets a user profile for accessing an Exchange folder for reports exported to Exchange.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
FormatDLLName	Returns the file name of the DLL corresponding to the export format.	Read only
FormatType	Returns/Sets CRExportFormatType (see table below) indicating the format type for the exported report (i.e., text, Excel, etc.)	Read/Write

<i>Constant</i>	<i>Value</i>
crEFTCharSeparatedValues	7
crEFTCommaSeparated-Values	5
crEFTCrystalReport	1
crEFTDataInterchange	2
crEFTExcel21	18
crEFTExcel30	19
crEFTExcel40	20
crEFTExcel50	21
crEFTExcel50Tabular	22
crEFTExplorer32Extend	25
crEFTHTML32Standard	24
crEFTLotus123WK3	13
crEFTLotus123WK1	12
crEFTLotus123WKS	11
crEFTNetScape20	26
crEFTNoFormat	0
crEFTODBC	23
crEFTPaginatedText	10
crEFTQuattroPro50	17
crEFTRecordStyle	3
crEFTRichText	4
crEFTTabSeparatedText	9
crEFTTabSeparatedValues	6
crEFTText	8
crEFTWordForDOS	15

Constant	Value	
crEFTWordForWindows	14	
crEFTWordPerfect	16	
Property	Description	Read/Write
HTMLFileName	Returns/set the HTML file name for reports exported to HTML format.	Read/Write
MailBccList	Returns/Sets a Blind Carbon Copy (BCC) list for reports e-mailed to a VIM e-mail account.	Read/Write
MailCcList	Returns/Sets a Carbon Copy (CC) list for reports e-mailed.	Read/Write
MailMessage	Returns/Sets the e-mail message included with e-mailed reports.	Read/Write
MailSubject	Returns/Sets the e-mail subject heading for reports being e-mailed.	Read/Write
MailToList	Returns/Sets the To list for reports being e-mailed.	Read/Write
NumberOfLinesPerPage	Returns/Sets the number of lines to appear per page of the report for report formats that are paginated. For example, HTML.	Read/Write
ODBCDataSourceName	Returns/Sets the ODBC data source for reports exported to ODBC.	Read/Write
ODBCDataSource-Password	Returns/Sets the password used to access an ODBC data source for reports exported to ODBC.	Read/Write
ODBCDataSourceUserID	Returns/Sets the user name used to access an ODBC data source for reports exported to ODBC.	Read/Write
ODBCExportTableName	Returns/Sets the database table in the ODBC data source that the report file exported to ODBC will be appended to. You can also create a new table using this property.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
UseReportDateFormat	Returns/Sets whether the date format used in the report should also be used in the exported report. Can be used for Data Interchange Format (DIF), Record Style Format, and comma, tab, or character separated format.	Read/Write
UseReportNumber-Format	Returns/Sets whether the number format used in the report should also be used in the exported report. Can be used for Data Interchange Format (DIF), Record Style Format, and comma, tab, or character separated format.	Read/Write

Methods

PromptForExportOptions

The **PromptForExport Options** method prompts the user for export information using default Crystal Report Engine dialog boxes.

Syntax

```
object.PromptForExportOptions
```

Reset

The **Reset** method clears all ExportOptions properties.

Syntax

```
object.Reset
```

FieldDefinitions Collection

The **FieldDefinitions** Collection is a collection of field definitions of all types. This collection is obtained from the DataFields property of the *GraphObject Object, Page 651*. The collection of fields represent the fields used to plot the values on a graph. Access a specific FieldDefinition object in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Count	Returns the number of field definitions contained by this collection.	Read only
Item	Returns specific FieldDefinition objects in this collection. Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)). The items in the collection are indexed in the order they are graphed.	Read only
Parent	Reference to the Parent object (<i>GraphObject Object, Page 651</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a field definition directly, for example, FieldDefinitions (1).

FieldObject Object

The **FieldObject** Object represents a field found in a report (i.e., special field, database field, parameter field, etc.). This object provides properties for retrieving information for a field in your report. A FieldObject Object is obtained from the Item property of the *ReportOptions Object, Page 697* (i.e., ReportObjects.Item(Index)), where the index references a special field, database field, parameter field, summary field, or group name field (except BLOB field - see *BlobField Object, Page 620*).

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Field	Returns the field definition object (depending on what the field object is, i.e., <i>DatabaseFieldDefinition Object, Page 624</i> , <i>FormulaFieldDefinition Object, Page 646</i> , <i>GroupNameFieldDefinition Object, Page 658</i> , <i>ParameterFieldDefinition Object, Page 673</i> , <i>SummaryFieldDefinition Object, Page 712</i> , and <i>SpecialVarFieldDefinition Object, Page 708</i>).	Read only
Font	Returns <i>Font Object, Page 644</i> , with information on what type of font is being used (i.e., name).	Read only
Kind	Returns CRObjectKind object which specifies what “kind” of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOleObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

FieldValue Object (32-bit only)

The **FieldValue** Object is only available through the *Window Object, Page 724*, event *DrillOnDetail, Page 727*, used in Visual Basic 5.0. When a user drill down on a Detail record value, all fields for that record are stored in an array of FieldValue objects that is returned in the *FieldValues* parameter of the *DrillOnDetail* event.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Name	Returns the name of the field.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
Value	Returns the data contained in the field.	Read only

Font Object

The **Font** Object provides properties for retrieving information and setting options for the font used in a specified field (i.e., Bold, line style, etc.). A Font Object is obtained from the *Font* property of the *FieldObject Object, Page 642*. Options, such as font name, defined by this object are dependent upon the printer driver selected for the report.

NOTE: You must always prefix when referencing the Font Object. Visual Basic includes a VB Font Object that is always included when creating a project, so conflicts will always occur unless the object is properly referenced. For example, CRPEAuto.Font. For more information see Object Naming Conflicts, Page 611.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Color	Returns/Sets a number representing the color of the font. The number represents an OLE color that can be set using RGB (i.e., RGB (255, 0, 0 for red) or by specifying CRCColor (see table below).	Read/Write
<hr/>		
<i>Constant</i>	<i>Value</i>	
crAqua	16776960	
crBlack	0	
crBlue	16711680	
crFuchsia	16711935	
crGray	8421504	
crGreen	32768	
crLime	65280	
crMaroon	128	
crNoColor	-1	
crOlive	32896	
crPurple	8388736	
crRed	255	
crSilver	12632256	
crTeal	8421376	
crWhite	16777215	
crYellow	65536	
<hr/>		
<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Italic	Returns/Sets Boolean value indicating whether or not the font is italicized.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns/Sets the name of the font. Available fonts for any report are dependent upon the printer driver selected for the report.	Read/Write
Parent	Reference to the Parent object (<i>FieldObject Object, Page 642</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
Size	Returns/Sets the point size of the font.	Read/Write
StrikeThrough	Returns/Sets Boolean value indicating whether or not the font is struck out.	Read/Write
Underline	Returns/Sets Boolean value indicating whether or not the font is underlined.	Read/Write
Weight	Returns/Sets the weight of the font. CRFontWeight (see table below) can be used for convenience.	Read/Write

<i>Constant</i>	<i>Value</i>
crFWBold	700
crFWDontCare	0
crFWExtraBold	800
crFWExtraLight	200
crFWHeavy	900
crFWLight	300
crFWMedium	500
crFNormal	400
crFWSemiBold	600
crFWThin	100

FormulaFieldDefinition Object

The **FormulaFieldDefinition** Object provides properties and methods for retrieving information and setting options for any formula field found in a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
FormulaFieldName	Returns the formula field name as it appears in the Formula Field list of the Formula Tab in the Insert Fields dialog box (i.e., ExampleFormula).. .	Read only
Kind	Returns CRFieldKind (see table below) which specifies what “kind” of field (i.e., database, summary, formula, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crDatabaseField	1
crFormulaField	2
crGroupNameField	5
crParameterField	6
crSpecialVarField	4
crSummaryField	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the formula field as it would be displayed (referenced) in the report (i.e., {@ExampleFormula}).	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Text	Returns/Sets the text of the formula. The formula text is changed immediately in the report. With that in mind, you should check the validity of the formula immediately after setting the Text property. Use <i>Check</i> , <i>Page 649</i> , to check formulas. If you generate a report with an invalid formula, you may receive an exception error. .	Read/Write
ValueType	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateFormat	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7
crOleField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12

<i>Constant</i>	<i>Value</i>
crTimeField	11
crTransientMemoField	13
crUnknownField	22

Methods

Check

The **Check** method checks formula for errors (i.e., syntax errors).

Syntax

```
object.Check
```

Returns

Boolean. TRUE if formula is valid; FALSE if formula contains errors. You can use LastErrorCode and LastErrorString properties from the *Report Object, Page 685*, in order to obtain details when FALSE is returned.

FormulaFieldDefinitions Collection

The **FormulaFieldDefinitions** Collection is a collection of named formulas in the report. Access a specific *FormulaFieldDefinition Object, Page 646*, in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Count	Returns the number of formula field definitions in the collection.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Item	Returns <i>FormulaFieldDefinition Object, Page 646</i> . Item has an index parameter that can be either a string reference to the formula field or a numeric, 1-based index. When a numeric index is used, the items in the collection are indexed in the order they were added to the report.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a formula field directly, for example, `FormulaFieldDefinitions("ExampleFormula")`.

GlobalOptions Object

The **GlobalOptions** Object provides properties for retrieving information or setting options for an application object. A GlobalOptions Object is obtained from the Options property of the *Application Object, Page 611*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
MorePrintEngineError-Messages	Sets Boolean value indicating more error messages at the application level. Use this object to turn off more error messages if they are switched on by default. It will not affect more print error messages at the report level.	Write only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Application Object, Page 611</i>).	Read only

GraphObject Object

The **GraphObject** Object represents a graph/chart found in a report. This object provides properties for retrieving information and setting options for a graph in your report (i.e., graph data type - group, detail or graph display type - bar, pie, etc.).

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
ColumnGroupNumber	Returns/Sets the group number that the column of the graph gets its data from (i.e., in a stacked bar graph, the “column” is the bar of the graph, so there will be one bar for each group).	Read/Write
ConditionField	Returns/Sets the Detail field to be graphed on. This property applies to only detail graphs; using this property with a cross-tab or group graph will result in an error.	Read/Write
CrossTabObject	Returns <i>CrossTabObject Object, Page 622</i> , with information regarding the cross-tab object used to graph on. This property applies only to cross-tab graphs; using this property with a detail or group graph will result in an error	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
DataFields	Returns the FieldDefinitions object that represents the data fields of the graph (i.e., numeric fields for graph data). Specifies a collection of graph data fields. This property applies only to detail graphs; using this property with a cross-tab or group graph will result in an error.	Read only
DataType	Returns CRGraphDataType (see table below) which specifies the type of data used in the graph.	Read only

<i>Constant</i>	<i>Value</i>
crCrossTabGraph	2
crDetailGraph	1
crGroupGraph	0

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Direction	Returns/Sets CRGraphDirection (see table below) indicating if the graph groups on only the rows of a cross-tab, only the columns, or a combination of both (i.e., for stacked bar graph). This property applies to only cross-tab graphs; using this property with a detail or group graph will result in an error.	Read/Write

<i>Constant</i>	<i>Value</i>
crGDColumnsOnly	1
crGDMixedColumnRow	3
crGDMixedRowColumn	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
DisplayType	Returns/Sets CRGraphDisplayType (see table below) indicating which graph representation type used for the graph (i.e., SideBySideGraph, PieGraph, etc.).	Read/Write

<i>Constant</i>	<i>Value</i>
crAreaGraph	120
crFaked3DPercentBarGraph	6
crFaked3DSideBySideBar-Graph	4
crFaked3DStackedBarGraph	5
crLineGraph	80
crMultiPieGraph	42
crPercentBarGraph	3
crPieGraph	40
crProportionalMultiPieGraph	43
crSideBySideGraph	0
crStackedBarGraph	2
crThreedBarGraph	160
crUnkownTypeGraph	1000
crUserDefinedGraph	500

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
FontFaceName	Returns/Sets the font for the title, subtitle, and footnote text included with the graph. Fonts available for any report are dependent upon the printer driver selected for the report.	Read/Write
FootNote	Returns/Sets the footnote text that appears at the bottom of the graph.	Read/Write
GroupsTitle	Returns/Sets the title of the groups which are being graphed.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Kind	Returns CRObjectKind object which specifies what "kind" of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOLEObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
MaxValue	Returns/Sets the maximum value that will appear in the graph. Any graph values above this value are not charted.	Read/Write
MinValue	Returns/Sets the minimum value that will appear in the graph. Any graph values below this value are not charted.	Read/Write
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
RowGroupNumber	Returns/Sets the group number of the group that supplies the data for the graphs rows. For example, in a stacked bar graph, the row would be the stacked data. For graphs with no rows, this value is -1. (This property does not apply to a detail graph.)	Read/Write

Property	Description	Read/Write
SeriesTitle	Returns/Sets the title of the series which is being graphed.	Read/Write
ShowDataValueEnabled	Returns/Sets Boolean value indicating whether or not to display the numeric value associated with each riser on the chart. If set to TRUE (1), a value appear in the graph for each riser.	Read/Write
ShowGridLineEnabled	Returns/Sets Boolean value indicating whether or not to display grid lines on the graph.	Read/Write
ShowLegendEnabled	Returns/Sets Boolean value indicating whether or not to display the graph legend.	Read/Write
ShowVerticalBarEnabled	Returns/Sets Boolean value indicating whether to display the bars in a bar graph vertically or horizontally.	Read/Write
SubTitle	Returns/Sets the subtitle text that will appear directly under the main title.	Read/Write
SummarizedField	Returns/Sets <i>SummaryFieldDefinition Object</i> , <i>Page 712</i> , used to supply the data for a group graph. (This property does not apply to a detail graph.)	Read/Write
Title	Returns/Sets the main title text that will appear above the graph.	Read/Write
XAxisTitle	Returns/Sets the text that will appear for the X axis. Not valid for Pie graphs.	Read/Write
YAxisTitle	Returns/Sets the text that will appear for the Y axis. Not valid for Pie graphs.	Read/Write
ZAxisTitle	Returns/Sets the text that will appear for the Z axis. This value is only valid for 3D graphs.	Read/Write

GroupAreaOptions Object

The **GroupAreaOptions** Object provides properties for retrieving information and setting options for a group area found in a report (i.e., keep group together, sort direction, etc.). A GroupAreaOptions Object is obtained from the GroupOptions property of the *Area Object, Page 615*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Condition	Returns/Sets CRGroupCondition (see table below) indicating when grouping should occur for Boolean or Date groups (i.e., for Boolean groups, group on change to TRUE; for date groups, group dates weekly, etc.).	Read/Write

<i>Constant</i>	<i>Value</i>
crGCAnnually	7
crGCAnyValue	14
crGCBiweekly	2
crGCDaily	0
crGCEveryNo	11
crGCEveryYes	10
crGCMonthly	4
crGCNextIsNo	13
crGCNextIsYes	12
crGCQuarterly	5
crGCSemiAnnually	6
crGCSemiMonthly	3
crGCToNo	9
crGCToYes	8
crGCWeekly	1

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ConditionField	Returns/Sets <i>DatabaseFieldDefinition Object</i> , <i>Page 624</i> , specifying the field that grouping occurs on.	Read/Write
DiscardOtherGroups	Returns/Sets Boolean value indicating of “other” groups in a group sort are discarded (i.e., those not included in a Top/Bottom N group sort when sorting groups by their summary field).	Read/Write
KeepGroupTogether	Returns/Sets Boolean value to keep group together (i.e., across page breaks).	Read/Write
NumberOfTopOrBottom-Groups	Returns/Sets the number of groups for a TopN group sort.	Read/Write
Parent	Reference to the Parent object (<i>Area Object, Page 615</i>).	Read only
RepeatGroupHeader	Returns/Sets Boolean value indicating if group headers should be repeated when group is split across pages.	Read/Write
Report	Reference to <i>Report Object, Page 685</i> .	Read only
SortDirection	Returns/Sets CRSortDirection (see table below) indicating the sort direction of the group name field, as opposed to the group summary field.	Read/Write

<i>Constant</i>	<i>Value</i>
crAscendingOrder	1
crDescendingOrder	0
crOriginalOrder	2
crSpecifiedOrder (Read only - this value cannot be set.)	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
TopOrBottomNGroups	Returns/Sets the value of CRCTopBottomNGroupSortOrder (see table below) indicating whether the which section of groups will be shown (i.e., TopN groups, BottomN groups, all groups, etc.).	Read/Write
<i>Constant</i>		<i>Value</i>
crAllGroupsSorted		1
crAllGroupsUnsorted		0
crBottomNGroups		3
crTopNGroups		2

GroupNameFieldDefinition Object

The **GroupNameFieldDefinition** Object provides properties and methods for retrieving information and setting options for a group name field found in a report (i.e., number of group, value type, etc.). A GroupNameFieldDefinition Object is obtained from the *Field* property of the *FieldObject Object*, *Page 642* when the specified field is a group name field.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
GroupNumber	Returns the group number of the specified group.	Read only
Kind	Returns CRFieldKind (see table below) which specifies what “kind” of field (i.e., database, summary, formula, etc.).	Read only
<i>Constant</i>		<i>Value</i>
crDatabaseField		1

<i>Constant</i>	<i>Value</i>
crFormulaField	2
crGroupNameField	5
crParameterField	6
crSpecialVarField	4
crSummaryField	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the group name field.	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
ValueType	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2

<i>Constant</i>	<i>Value</i>
crNumberField	7
crOLEField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

LineObject Object

The **LineObject** Object represents a line drawn on a report. This object provides properties for getting information for lines on a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjektKind (see table below) which specifies what “kind” of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOLEObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

OLEObject Object

The **OLEObject** Object represents an OLE object in a report. This object provides properties for getting information for OLE objects in a report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjectKind (see table below) which specifies what “kind” of object (i.e., box, cross-tab, field, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOleObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Report	Reference to <i>Report Object, Page 685.</i>	Read only

Page Object (32-bit only)

The **Page** Object is part of the Page Engine. Use the Page Engine when designing web sites using Active Server Pages, the Crystal Report Engine Automation Server, and the Crystal Design-Time ActiveX Control. For complete information on using the automation server and ActiveX control to build web sites, see *Building Active Web Sites, Page 25*. Unless you are experienced with the Crystal Report Engine Object Library, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code in your Active Server Pages that controls the Page Engine objects.

The Page Engine generates pages of a report on the web server and sends the pages to client web browsers as they are requested. For example, when a user first requests a report, only the first page is sent to the web browser. If the user pages forward or backward in the report, or requests a specific page, only that page is sent. This limits the resources required by the web server and reduces download time for the client browser. A Page Object is a single generated page that is sent to the browser.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
IsLastPage	Returns Boolean value indicating whether the page generated is the last page of the report.	Read only
PageNumber	Returns the page number of the page generated.	Read only
Parent	Reference to the Parent object (<i>PageGenerator Object (32-bit only), Page 667</i>).	Read only
Report	Reference to <i>Report Object, Page 685.</i>	Read only

Methods

RenderEPF

The **RenderEPF** method returns a variant that contains EPF code for the report page.

Syntax

```
variantdata = object.RenderEPF (ResultType as  
CRRenderResultType)
```

Parameters

Parameter	Description
ResultType	Returns CRRenderResultType (see table below) indicating whether the page will be rendered using strings or arrays.

Constant	Value
crBSTRType	8
crUISafeArrayType	8209

RenderHTML

The **RenderHTML** method returns a variant that contains HTML code for the report page.

Syntax

```
variantdata = object.RenderHTML (IncludeDrillDownLinks as  
Boolean, PageStyle as CRHTMLPageStyle, ToolbarStyle as  
CRHTMLToolbarStyle, baseURL as string, ResultType as  
CRRenderResultType)
```

Parameters

Parameter	Description
IncludeDrillDown-Links	Indicates whether or not the HTML page will include hyperlinks for drilling-down on summary data.
PageStyle	Specifies CRHTMLPageStyle (see table below) indicating the style of the HTML page to be rendered.

Constant	Value
crFramePageStyle	2

<i>Parameter</i>	<i>Description</i>
ToolbarStyle	Specifies CRHTMLToolbarStyle (see table below) indicating the style of the toolbar to be used.
<i>Constant</i>	<i>Value</i>
crToolbarRefreshButton	1
crToolbarSearchBox	2
<i>Parameter</i>	<i>Description</i>
BaseUrl	The URL used to access the report when it is first generated.
ResultType	Returns CRRenderResultType (see table below) indicating whether the page will be rendered using strings or arrays.
<i>Constant</i>	<i>Value</i>
crBSTRType	8
crUISafeArrayType	8209

PageEngine Object (32-bit only)

Use the **PageEngine** object when designing web sites using Active Server Pages, the Crystal Report Engine Automation Server, and the Crystal Design-Time ActiveX Control. For complete information on using the automation server and ActiveX control to build web sites, see *Building Active Web Sites, Page 25*. Unless you are experienced with the Crystal Report Engine Object Library, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code in your Active Server Pages that controls the Page Engine objects.

The Page Engine generates pages of a report on the web server and sends the pages to client web browsers as they are requested. For example, when a user first requests a report, only the first page is sent to the web browser. If the user pages forward or backward in the report, or requests a specific page, only that page is sent. This limits the resources required by the web server and reduces download time for the client browser.

Properties

Property	Description	Read/Write
ImageOptions	Returns/Sets CRImageType (see table below) indicating the image type for EPF format.	Read/Write
Constant		Value
crDIBImageType		1
crJPEGImageType		2

Methods

CreatePageGenerator

The **CreatePageGenerator** method returns a *PageGenerator Object (32-bit only)*, [Page 667](#), allowing you to get pages from the view of the report, specified by the GroupPath parameter.

Syntax

```
PgGenObject = object.CreatePageGenerator (GroupPath)
```

Parameters

Parameter	Description
GroupPath	Specifies an integer array that represents the group path (separated by "/"). An empty array would represent the entire report, an array (0, 1) would represent a drill down on the second group member within the first group member (i.e., 0 = first group in Group #1 and 1 = second group in Group #2).

NOTE: Another optional parameter, DrillDownLevel, will appear in the Object Browser for the CreatePageGenerator Method. This parameter has not been implemented; the value will be ignored.

RenderTotallerETF

The **RenderTotallerETF** method returns a variant that contains ETF code for the Group Tree.

Syntax

```
variantData = object.RenderTotallerETF (RootGroupPath,  
StartingChildNumber, PastRootLevels, MaxNodeCount, ResultType)
```

Parameters

Parameter	Description
MaxNodeCount	The maximum number of nodes for the Group Tree.
PastRootLevels	An array of past root levels.
ResultType	Returns CRRenderResultType (see table below) indicating whether the page will be rendered using strings or arrays.

Constant	Value
crBSTRType	8
crUISafeArrayType	8209

Parameter	Description
RootGroupPath	Specifies an integer array that represents the root group path (separated by "/"). An empty array would represent the entire report, an array (0, 1) would represent a drill down on the second group member within the first group member (i.e., 0 = first group in Group #1 and 1 = second group in Group #2).
StartingChild-Number	The child level to display the report grouping at.

RenderTotallerHTML

The **RenderTotallerHTML** method returns a variant that contains HTML code for the Group Tree.

Syntax

```
variantData = object.RenderTotallerHTML (RootGroupPath,  
StartingChildNumber, PastRootLevels, MaxNodeCount,  
OpenGroupPath, IncludeDrillDownLinks, baseURL, ResultType)
```

Parameters

Parameter	Description
BaseURL	The URL address of the report when it is first generated by the web server.
IncludeDrillDown-Links	Indicates whether or not drill-down hyperlinks are generated for summary values in the report.

<i>Parameter</i>	<i>Description</i>
MaxNodeCount	The maximum number of nodes to display in the Group Tree.
OpenGroupPath	An array of groups to be opened in the report.
PastRootLevels	A value indicating the number of past root levels.
ResultType	Returns CRRenderResultType (see table below) indicating whether the page will be rendered using strings or arrays.

<i>Constant</i>	<i>Value</i>
crBSTRType	8
crUISafeArrayType	8209

<i>Parameter</i>	<i>Description</i>
RootGroupPath	Specifies an integer array that represents the root group path (separated by "/"). An empty array would represent the entire report, an array (0, 1) would represent a drill down on the second group member within the first group member (i.e., 0 = first group in Group #1 and 1 = second group in Group #2).
StartingChild-Number	Long

PageGenerator Object (32-bit only)

The **PageGenerator** Object is part of the Page Engine. Use the Page Engine when designing web sites using Active Server Pages, the Crystal Report Engine Automation Server, and the Crystal Design-Time ActiveX Control. For complete information on using the automation server and ActiveX control to build web sites, see *Building Active Web Sites*, *Page 25*. Unless you are experienced with the Crystal Report Engine Object Library, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code in your Active Server Pages that controls the Page Engine objects.

The Page Engine generates pages of a report on the web server and sends the pages to client web browsers as they are requested. For example, when a user first requests a report, only the first page is sent to the web browser. If the user pages forward or backward in the report, or requests a specific page, only that page is sent. This limits the resources required by the web server and reduces download time for the client browser. A PageGenerator object generates *Page Object (32-bit only)*, *Page 662*, as they are requested and allows options for manipulating the report as a whole.

Properties

Property	Description	Read/Write
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
GroupPath	Returns the GroupPath parameter set by <i>CreatePageGenerator, Page 665</i> .	Read only
Pages	Returns <i>Pages Collection (32-bit only), Page 670</i> , which is a collection of pages in the report.	Read only
Parent	Reference to the Parent object (<i>PageEngine Object (32-bit only), Page 664</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Methods

DrillOnGraph

The **DrillOnGraph** method creates a new *PageGenerator Object (32-bit only), Page 667*, that results from drilling down on the specified point in a graph on the given page.

Syntax

```
pgGenObject = object.DrillOnGraph (PageNumber, XOffset, YOffset)
```

Parameters

Parameter	Description
PageNumber	Specifies the page number.
XOffset	Specifies the X coordinate on page in twips where the drill down occurred.
YOffset	Specifies the Y coordinate on page in twips where the drill down occurred.

GetPageNumberForGroup

The **GetPageNumberForGroup** method returns the page number on which the given group starts.

Syntax

```
numbervar = object.GetPageNumberForGroup (GroupPath)
```

Parameters

Parameter	Description
GroupPath	Specifies an integer array that represents the group path (separated by "/"). An empty array would represent the entire report, an array (0, 1) would represent a drill down on the second group member within the first group member (i.e., 0 = first group in Group #1 and 1 = second group in Group #2).

SearchForText

The **SearchForText** method searches for the given text, starting on the given page. If the text is found, the PageNumber is set to the number of the page on which it was found. Returns a Boolean value indicating if text was found.

Syntax

```
Boolean = object.SearchForText (Text, Direction, PageNumber)
```

Parameters

Parameter	Description
Direction	Specifies CRSearchDirection (see table below) indicating the direction to sort in.

Constant	Value
crAscendingOrder	1
crDescendingOrder	0
crOriginalOrder	2

Parameter	Description
PageNumber	Specifies the page the start searching in and returns the page number of the first occurrence of the sought after text.
Text	Specifies the text string being searched for.

Pages Collection (32-bit only)

The **Pages Collection** is part of the Page Engine. Use the Page Engine when designing web sites using Active Server Pages, the Crystal Report Engine Automation Server, and the Crystal Design-Time ActiveX Control. For complete information on using the automation server and ActiveX control to build web sites, see *Building Active Web Sites, Page 25*. Unless you are experienced with the Crystal Report Engine Object Library, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code in your Active Server Pages that controls the Page Engine objects.

The Page Engine generates pages of a report on the web server and sends the pages to client web browsers as they are requested. For example, when a user first requests a report, only the first page is sent to the web browser. If the user pages forward or backward in the report, or requests a specific page, only that page is sent. This limits the resources required by the web server and reduces download time for the client browser.

The **Pages Collection** is a collection of *Page Object (32-bit only), Page 662*. Access a specific Page Object in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Count	Returns the number of page objects in the collection.	Read only
Item	Returns <i>Page Object (32-bit only), Page 662</i> . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)).	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a page directly, for example, Pages (1).

PageSetup Object

The **PageSetup** Object provides properties for retrieving information and setting options for page setup in a report (i.e., paper size, margins, etc.). A PageSetup Object is obtained from the PageSetup property of the *Report Object, Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
BottomMargin	Returns/Sets value specifying the bottom margin.*	Read/Write
LeftMargin	Returns/Sets value specifying the left margin.*	Read/Write
PaperOrientation	Returns/Sets CRPaperOrientation (see table below) indicating the paper orientation.	Read/Write

<i>Constant</i>	<i>Value</i>
crDefaultPaperOrientation	0
crLandscape	2
crPortrait	1

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
PaperSize	Returns/Sets CRPaperSize (see table below) indicating the paper size. Check your printer documentation for a list of paper sizes available for use with your printer. If you choose an unavailable paper size, the default (letter) size will be used.	Read/Write

<i>Constant</i>	<i>Value</i>
crDefaultPaperSize	0

<i>Constant</i>	<i>Value</i>
crPaper10x14	16
crPaper11x17	17
crPaperA3	8
crPaperA4	9
crPaperA4Small	10
crPaperA5	11
crPaperB4	12
crPaperB5	13
crPaperCsheet	24
crPaperDsheet	25
crPaperEnvelope10	20
crPaperEnvelope11	21
crPaperEnvelope12	22
crPaperEnvelope14	23
crPaperEnvelope9	19
crPaperEnvelopeB4	33
crPaperEnvelopeB5	34
crPaperEnvelopeB6	35
crPaperEnvelopeC3	28
crPaperEnvelopeC4	29
crPaperEnvelopeC5	30
crPaperEnvelopeC6	31
crPaperEnvelopeC65	32
crPaperEnvelopeDL	27
crPaperEnvelopeItaly	36
crPaperEnvelopeMonarch	37
crPaperEnvelopePersonal	38
crPaperEsheet	26
crPaperExecutive	7
crPaperFanfoldLegalGerman	41
crPaperFanfoldStdGerman	40
crPaperFanfoldUS	39
crPaperFolio	14
crPaperLedger	4

<i>Constant</i>	<i>Value</i>
crPaperLegal	5
crPaperLetter	1
crPaperLetterSmall	2
crPaperNote	18
crPaperQuarto	15
crPaperStatement	6
crPaperTabloid	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
RightMargin	Returns/Sets value specifying the right margin.*	Read/Write
TopMargin	Returns/Sets value specifying the top margin.*	Read/Write

*Left, right, top, and bottom values are all in twips. A twip is 1/1440 of an inch; there are 20 twips in a point. To set .5" margins, for example, you would enter the value 720.

NOTE: Left, Right, Top, and Bottom are set to -1 by default. Unless otherwise specified, the default margins set in the report are used.

ParameterFieldDefinition Object

The **ParameterFieldDefinition** Object represents a parameter field in the report. This object provides properties and methods for retrieving information and setting options for a parameter field in your report (i.e., current value, default value, etc.). A ParameterFieldDefinition Object is obtain from the Field property of the *Area Object, Page 615* when the specified field is a parameter field.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
CurrentValue	Returns the value entered by the user or when set using SetCurrentValue. If the user clicked Cancel or no value is set, a null is returned.	Read only
CurrentValueSet	Boolean value that will equal TRUE (1) if the user enters a value at the prompt or a value has been set using SetCurrentValue and FALSE (0) if the user clicks Cancel or no value has been set.	Read only
DefaultValue	Returns the default value assigned to the parameter field if one was set. If DefaultValueSet is FALSE, a null is returned.	Read only
DefaultValueSet	Returns Boolean value indicating whether or not a default value was set for the parameter field when the parameter field was created or modified in Seagate Crystal Reports. The value can be either TRUE (1) if the field was given a default value or FALSE (0) if it was not.	Read only
Kind	Returns CRFieldKind (see table below) which specifies what "kind" of field (database, summary, formula, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crDatabaseField	1
crFormulaField	2
crGroupNameField	5
crParameterField	6
crSpecialVarField	4
crSummaryField	3

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the parameter field as it appears in the Parameter Field list of the Parameter Tab in the Insert Fields dialog box (i.e., ExampleParameter).	Read only
NeedsCurrentValue	Returns Boolean value indicating if the parameter needs a value entered (no current value is set). TRUE if no current value is set; FALSE if current value is set.	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
ParameterFieldName	Returns the name of the parameter field as it is displayed (referenced) in the report (i.e., {?ExampleParameter}).	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Prompt	Returns/Sets the prompting text, if any, that appears when the user runs the report for the first time or refreshes the data. This will be either the prompt assigned by the user when the parameter field was inserted into the report or the prompt that was set from code.	Read/Write
Report	Reference to <i>Report Object, Page 685</i> .	Read only
ReportName	Returns the subreport name if the parameter is part of a subreport; otherwise, a null is returned.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ValueType	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only
<i>Constant</i>		<i>Value</i>
crBitmapField		17
crBlobField		15
crBooleanField		9
crChartField		21
crCurrencyField		8
crDateField		10
crDateTimeField		16
crIconField		18
crInt16sField		3
crInt16uField		4
crInt32sField		5
crInt32uField		6
crInt8sField		1
crInt8uField		2
crNumberField		7
crOleField		20
crPersistentMemoField		14
crPictureField		19
crStringField		12
crTimeField		11
crTransientMemoField		13
crUnknownField		22

Methods

SetCurrentValue

The **SetCurrentValue** method sets the current value of the parameter. This is the value that will be used for the parameter in the report. If the current value is set, no prompt will appear for the parameter when the report is run.

Syntax

```
object.SetCurrentValue CurrentValue, ValueType
```

Parameters

Parameter	Description
CurrentValue	Specifies the value you want to set the parameter field current value to. Can be number, currency, date, string, or Boolean.
ValueType (Optional)	Specifies CRFieldValueType (see table below) indicating the type of the current value (i.e., number, currency, etc.). You may either change the parameter field type using this parameter or keep the current type (as specified in the report) by omitting this parameter.

Constant	Value
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7

<i>Constant</i>	<i>Value</i>
crOLEField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

SetDefaultValue

The **SetDefaultValue** method sets the default value for the parameter. This is the value that will be used as the default shown when prompting for a value.

Syntax

```
object.SetDefaultValue DefaultValue, ValueType
```

Parameters

<i>Parameter</i>	<i>Type</i>
DefaultValue	Specifies the value you want to set the parameter field default value to. Can be number, currency, date, string, or Boolean.
ValueType (Optional)	Specifies CRFieldValueType (see table below) indicating the type of the default value (i.e., number, currency, etc.). You may either change the parameter field type using this parameter or keep the current type (as specified in the report) by omitting this parameter.

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4

<i>Constant</i>	<i>Value</i>
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7
crOleField	20
crPersistentMemoField	14
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

ParameterFieldDefinitions Collection

The **ParameterFieldDefinitions** Collection is a collection of parameter fields in the report. If the report contains any subreports, parameter fields in the subreports will also be included in the collection. Access a specific *ParameterFieldDefinition Object*, *Page 673*, in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Count	Returns the number of parameter fields in the collection.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Item	Returns <i>ParameterFieldDefinition Object, Page 673</i> , in a collection. This property has two parameters: Index and ReportName (optional). The index can be numeric (1-based), for example, Item(1). When a numeric index is used, the items in the collection are indexed in the order they were added to the report followed by the parameter fields from any subreport, in the same order. The second parameter is not necessary if a numeric index is used. The index can also be a string, representing the parameter name. If the report name is not included, the parameter field is assumed to be in the primary report. If the parameter field exists in a subreport, you must also pass the subreport name as the second parameter for Item.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a parameter field directly, for example, ParameterFieldDefintions(1).

PrinterInfo Object

The **PrinterInfo** Object provides properties for retrieving information and setting options for the specified printer used to print a report (i.e., driver name, port name, etc.). A PrinterInfo Object is obtained from the PrinterInfo property of the *Report Object, Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
DriverName	Returns the name of the printer driver for the printer that is selected in the report. If the default printer is used, no data is returned.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
PortName	Returns the name of the printer port. If the default printer is used, no data is returned.	Read only
PrinterName	Returns the name of the printer. If the default printer is used, no data is returned.	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

PrintingStatus Object

The **PrintingStatus** Object provides properties for retrieving information and setting options for the printing status of a report (i.e., number of pages, latest page to be printed, etc.). A PrintingStatus Object is obtained from the PrintingStatus property of the *Report Object, Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
LastestPageNumber	Returns the number of the last page that's currently available. Once the printing is complete, this value is the number of the last page. You will receive an incorrect result if you try to obtain this value immediately after using a method such as <i>ShowNextPage, Page 722</i> , that causes a change to the preview window. Call the Visual Basic function, DoEvents (see Visual Basic documentation for more information) before requesting the latest page number.	Read only
NumberOfPages	Returns the number of pages in the report.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Progress	Returns CRPrintingProgress (see table below) indicating the status of printing report (i.e., cancelled, in progress, not started, etc.). Corresponds to <i>PEJobInfo, Page 423</i> , and <i>PEGetJobStatus, Page 171</i> .	Read only

<i>Constant</i>	<i>Value</i>
crPrintingCancelled	5
crPrintingCompleted	3
crPrintingFailed	4

<i>Constant</i>	<i>Value</i>
crPrintingHalted	6
crPrintingInProgress	2
crPrintingNotStarted	1

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
RecordsPrinted	Returns the number of records actually printed. You will receive an incorrect result if you try to obtain this value immediately after using a method such as <i>ShowNextPage</i> , <i>Page 722</i> , that causes a change to the preview window. Call the Visual Basic function, DoEvents (see Visual Basic documentation for more information) before requesting the records printed.	Read only
RecordsRead	Returns the number of records actually processed.	Read only
RecordsSelected	Returns the number of records selected for inclusion in the report out of the total number of records read.	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only
StartPageNumber	Returns the number of the starting page of preview or printing.	Read only

PrintWindowOptions Object

The **PrintWindowOptions** Object provides properties for retrieving information and setting options for the print window (i.e., available buttons, Group Tree, etc.). Many of the properties in this object must be set prior to enabling other events. For example, before enabling the *CancelButtonClicked*, *Page 725* event from the Window Object, you must first set the HasCancelButton property from this object to TRUE. A PrintWindowOptions Object is obtained from the PrintWindowOptions property of the *Report Object*, *Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
CanDrillDown	Returns/Sets Boolean value indicating whether you can drill down on previewed report.	Read/Write
HasCancelButton	Returns/Sets Boolean value indicating whether the preview window has a Cancel button.	Read/Write
HasCloseButton	Returns/Sets Boolean value indicating whether the preview window has a Close button.	Read/Write
HasExportButton	Returns/Sets Boolean value indicating whether the preview window has an Export button.	Read/Write
HasGroupTree	Returns/Sets Boolean value indicating if the preview window includes the Group Tree. You must also set the HasGroupTree property to TRUE in the <i>ReportOptions Object, Page 697</i> , to have the Group Tree show in the preview window.	Read/Write
HasNavigationControls	Returns/Sets Boolean value indicating whether the preview window has navigation controls for moving through report pages.	Read/Write
HasPrintButton	Returns/Sets Boolean value indicating whether the preview window has a Print button.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
HasPrintSetupButton	Returns/Sets Boolean value indicating whether the preview window has a Print Setup button.	Read/Write
HasProgressControls	Returns/Sets Boolean value indicating whether the preview window has progress controls.	Read/Write
HasRefreshButton	Returns/Sets Boolean value indicating whether the preview window has a Refresh button.	Read/Write
HasSearchButton	Returns/Sets Boolean value indicating whether the preview window has a Search button.	Read/Write
HasZoomControl	Returns/Sets Boolean value indicating whether the preview window has zoom controls for changing the magnification of the report.	Read/Write
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
TrackCursorInfo	Returns <i>TrackCursorInfo Object, Page 717</i> .	Read only

Report Object

A report corresponds to a print job in the Crystal Report Engine. When the report object is destroyed, or goes out of focus, it closes the print job. It holds on to *Application Object, Page 611*. When a Report Object get destroyed, it releases the application.

Access to the Report Object is dependent on the object variable you create. If the object variable goes out of scope, you will lose access to the Report Object and, therefore, the report. You may want to declare your Report Object variable as Global.

Properties

<i>Properties</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Areas	Returns <i>Areas Collection</i> , <i>Page 619</i> , a collection of all the areas in the report which can be indexed by a number or by a string, such as "RH", "GF1". The areas are in the same order as on the Seagate Crystal Reports Design Tab. For Example: RH, PH, GH1,...GHn, D, GFn,...GF1, RF, PF. The abbreviations for areas are case sensitive.	Read only
Database	Returns the <i>Database Object</i> , <i>Page 623</i> , which represents the database used in the report.	Read only
DialogParentWindow	Sets the parent window of report dialog boxes. If not set, the parent window is the preview window.	Write only
EventInfo	Returns <i>EventInfo Object (32-bit only)</i> , <i>Page 636</i> .	Read only
ExportOptions	Returns <i>ExportOptions Object</i> , <i>Page 637</i> .	Read only
FormulaFields	Returns <i>FormulaFieldDefinitions Collection</i> , <i>Page 649</i> , a collection of all the named FormulaFieldDefinitions defined in the Report.	Read only
GroupSelectionFormula	Returns/Sets the group selection formula.	Read/Write
GroupSortFields	Returns <i>SortFields Collection</i> , <i>Page 707</i> , a collection of group sort fields.	Read only
HasSavedData	Returns Boolean value indicating whether report includes saved data.	Read only

<i>Properties</i>	<i>Description</i>	<i>Read/Write</i>
LastErrorCode	Returns last error code for report-level errors.	Read only
LastErrorString	Returns last error string for report-level errors.	Read only
NumberOfGroup	Returns the number of groups in the report.	Read only
Options	Returns <i>ReportOptions Object</i> , <i>Page 697</i> .	Read only
PageEngine	Returns <i>PageEngine Object (32-bit only)</i> , <i>Page 664</i> .	Read only
PageSetup	Returns <i>PageSetup Object</i> , <i>Page 671</i> .	Read only
ParameterFields	Returns <i>ParameterFieldDefinitions Collection</i> , <i>Page 679</i> , a collection of all the ParameterFieldDefinitions defined in the report. This property will return parameter field found in the main report as well as any subreports included in the report (i.e., if the main report has 3 parameters and a subreport included within the report has an additional 2 parameters, the number of parameter fields in the collection returned by Report.ParameterFields would be 5).	Read only

<i>Properties</i>	<i>Description</i>	<i>Read/Write</i>
ParameterPromptingEnabled	Returns/Sets Boolean value indicating whether prompting dialog box for parameter field information is used. <ul style="list-style-type: none"> ● If current values of all parameters are set, no prompting will occur for parameters, even if this is set to TRUE. ● If current values of all parameters are not set, prompting will occur for parameters that need a current value if this is set to FALSE. 	Read/Write
Parent	Reference to the Parent object (<i>Application Object, Page 611</i>).	Read only
PrintDate	Returns/Sets the print date for the report. By default, the current date will be used.	Read/Write
PrinterInfo	Returns <i>PrinterInfo Object, Page 681</i> .	Read only
PrintingStatus	Returns <i>PrintingStatus Object, Page 681</i> .	Read only
PrintWindowOptions	Returns <i>PrintWindowOptions Object, Page 683</i> .	Read only
ProgressDialogEnabled	Sets Boolean value indicating whether or not to enable the progress dialog box when printing or exporting.	Write only
RecordSelectionFormula	Returns/Sets record selection formula.	Read/Write
RecordSortFields	Returns <i>SortFields Collection, Page 707</i> , a collection of record sort fields.	Read only
ReportSummaryInfo	Returns <i>ReportSummaryInfo Object, Page 699</i> .	Read only
Sections	Returns <i>Sections Collection, Page 705</i> , a collection of all the sections in the report.	Read only

<i>Properties</i>	<i>Description</i>	<i>Read/Write</i>
SQLQueryString	Returns/Sets the SQL query used in the report.	Read/Write
SummaryFields	Returns <i>SummaryFieldDefinitions Collection</i> , <i>Page 715</i> , for group and report summaries (cross-tab summaries not available using this property).	Read only

Methods

AddGroup

The **AddGroup** method adds a group to the report. ConditionField indicates the field for grouping, Condition indicates a change in a field value that generates a grouping, and SortDirection specifies the direction in which groups are sorted.

Syntax

```
object.AddGroup GroupN, ConditionField, Condition, SortDirection
```

Parameters

<i>Parameter</i>	<i>Description</i>
GroupN	Specifies the number of the group to be added (the position of the group in relation to existing groups). For example, to add a group to the first position, set GroupN to 1.
ConditionField	Specifies the field to be grouped. The field can be a database field definition object or the field name.
Condition	Specifies CRGroupCondition (see table below) indicating the grouping condition (i.e., group on any value).

<i>Constant</i>	<i>Value</i>
crGCAannually	7
crGCAnyValue	14
crGCBiweekly	2
crGCDaily	0
crGCEveryNo	11
crGCEveryYes	10

<i>Constant</i>	<i>Value</i>
crGCMonthly	4
crGCNextIsNo	13
crGCNextIsYes	12
crGCQuarterly	5
crGCSEmiAnnually	6
crGCSemimonthly	3
crGCToNo	9
crGCToYes	8
crGCWeekly	1

<i>Parameter</i>	<i>Description</i>
SortDirection	Specifies CRSortDirection (see table below) indicating the sort direction for the group (i.e., ascending, descending).

<i>Constant</i>	<i>Value</i>
crAscendingOrder	1
crDescendingOrder	0
crOriginalOrder	2

CancelPrinting

The **CancelPrinting** method cancels the printing of a report. It corresponds to *PECancelPrintJob*, *Page 121*, of the Crystal Report Engine.

Syntax

```
object.CancelPrinting
```

ClearError

The **ClearError** method clears the last error code or last error string at the report level.

Syntax

```
object.ClearError
```

DiscardSavedData

The **DiscardSavedData** method discards any saved data with the report before previewing. It corresponds to *PEDiscardSavedData*, *Page 140*, of the Crystal Report Engine.

Syntax

```
object.DiscardSavedData
```

Export

The **Export** method exports reports to a format and destination specified with *ExportOptions Object*, [Page 637](#).

Syntax

```
object.Export PromptUser
```

Parameters

Parameter	Description
PromptUser (Optional)	Specifies Boolean value indicating if user should be prompted for export options (if you don't want to prompt). All necessary export options have to be set or prompt will be used whether set this parameter is set to TRUE or FALSE.

OpenSubreport

Every subreport has a pointer to its parent report. Subreports hold on to their parent reports until they are destroyed. The **OpenSubreport** method opens a subreport contained in the report and returns a *Report Object*, [Page 685](#), corresponding to the named subreport. It corresponds to *PReportEngine::OpenSubreport*, [Page 250](#), of the Crystal Report Engine.

Syntax

```
set reportvar = object.OpenSubreport ( SubreportName )
```

Parameters

Parameter	Description
SubreportName	Specifies the file name of the subreport to be opened.

Returns

Report

Preview

The **Preview** method reviews the subreport and returns the *View Object*, [Page 719](#).

Syntax

```
Set ViewVar = object.Preview (Title, Left, Top, Width, Height,  
Style, ParentWindow)
```

to capture the View object or,

```
object.Preview Title, Left, Top, Width, Height, Style,  
ParentWindow
```

for use without capturing the View object.

Parameters

Parameter	Description
Title (Optional)	Specifies the string that contains the title you want to appear on the title bar if you are printing the report to a window.
Left (Optional)	Specifies the x coordinate of the upper left hand corner of the print window, in pixels.
Top (Optional)	Specifies the y coordinate of the top of the print window, in pixels.
Width (Optional)	Specifies the width of the print window, in pixels.
Height (Optional)	Specifies the height of the print window, in pixels
Style (Optional)	Specifies the style of the window being created. Style settings can be combined using the bitwise "OR" operator. Select a style from the list that appears with <i>PEOutputToWindow</i> , <i>Page 257</i> .
Parent Window (Optional)	Specifies the handle to the Parent Window if the print window is a child of that window.

Returns

The *View Object*, *Page 719*, which represents the view created by previewing.

PrintOut

The **PrintOut** method prints out the specified pages of the report to the printer selected using the *SelectPrinter*, *Page 693* method. If no printer is selected, the default printer specified in the report will be used.

Syntax

```
object.PrintOut PromptUser, NumberOfCopies, Collated,  
StartPageNumber, StopPageNumber
```

Parameters

Parameter	Description
PromptUser (Optional)	Specifies Boolean value indicating if the user should be prompted for printer options.
NumberOfCopies (Optional)	Specifies the number of report copies you want printed.
Collated (Optional)	Specifies Boolean value specifying whether or not you want the report copies collated.
StartPageNumber (Optional)	Specifies the first page you want printed.
StopPageNumber (Optional)	Specifies the last page you want printed.

ReadRecords (32-bit only)

The **ReadRecords** method reads records in the report.

Syntax

```
object.ReadRecords
```

SelectPrinter

The **SelectPrinter** method selects a different printer for the report. It corresponds to *PESelectPrinter*, *Page 267*, of the Crystal Report Engine.

Syntax

```
object.SelectPrinter DriverName, PrinterName, PortName
```

Parameters

Parameter	Description
DriverName	String that contains the name of the printer driver for the selected printer. See <i>DriverName</i> property of <i>PrinterInfo Object</i> , <i>Page 681</i> for more information.
PrinterName	String that contains the printer name for the selected printer. See <i>PrinterName</i> property of <i>PrinterInfo Object</i> , <i>Page 681</i> for more information.
PortName	String that contains the port name for the port to which the selected printer is attached. See <i>PortName</i> property of <i>PrinterInfo Object</i> , <i>Page 681</i> for more information.

Events

ReadingRecords

The **ReadingRecords** event occurs when records are read (i.e., when a report is previewed, printed, refreshed, etc.). Reading occurs at random intervals so a different number of reading record events may occur each time the same report is previewed, printed, etc.

Syntax

```
Event ReadingRecords (ReadRecords As Long, RecordsSelected As  
Long, Cancelled As Boolean, Done As Boolean)
```

Parameters

Parameter	Description
Cancelled	Specifies Boolean value indicating the record reading was cancelled.
Done	Specifies Boolean value indication whether record reading is finished.
RecordsRead	Specifies the number of records read to process the report.
RecordsSelected	Specifies the number of records selected to appear in the report.
ReportName	Reserved for future use.

Remarks

This event is enabled using the *ReadingRecordsEventEnabled* property of the *EventInfo Object* (32-bit only), Page 636, obtained through the *EventInfo* property of the *Report Object*, Page 685.

NOTE: For more information on events, see *Object Library Events*, Page 611.

Start

The **Start** event occurs when the report starts printing or previewing.

Syntax

```
Event Start (Destination As CRPrintingDestination,  
useDefault As Boolean)
```

Parameters

<i>Parameter</i>	<i>Description</i>
Destination	Specifies CRPrintingDestination (see table below) indicating what “destination” the report will be printed to (i.e., printer, window, exported, etc.).

<i>Constant</i>	<i>Value</i>
crFromQuery	4
crToExport	3
crToNoWhere	0
crToPrinter	2
crToWindow	1

<i>Parameter</i>	<i>Description</i>
useDefault	Specifies Boolean value indicating if default behavior should occur after users event code has run (i.e., should the report start to print/preview).

Remarks

This event is enabled using the StartStopEventEnabled property of the *EventInfo Object* (32-bit only), *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see *Object Library Events*, *Page 611*.

Stop

The **Stop** event that occurs when report stops printing/previewing.

Syntax

```
Event Stop (Destination As CRPrintingDestination,  
Status As CRPrintingProgress)
```

Parameters

<i>Parameter</i>	<i>Description</i>
Destination	Specifies CRPrintingDestination (see table below) indicating what “destination” the report will be printed to (i.e., printer, window, exported, etc.).

<i>Constant</i>	<i>Value</i>
crFromQuery	4

<i>Constant</i>	<i>Value</i>
crToExport	3
crToNoWhere	0
crToPrinter	2
crToWindow	1

<i>Parameter</i>	<i>Description</i>
Status	Specifies CRPrintingProgress (see table below) indicating the progress of the report print job (i.e., in progress, halted, cancelled, etc.)

<i>Constant</i>	<i>Value</i>
crPrintingCancelled	5
crPrintingCompleted	3
crPrintingFailed	4
crPrintingHalted	6
crPrintingInProgress	2
crPrintingNotStarted	1

Remarks

This event is enabled using the StartStopEventEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see Object Library Events, Page 611.

ReportObjects Collection

The **ReportObjects** Collection is a collection of report objects in a section. Report objects can be a field, text, ole, cross-tab, subreport, BLOB field, Box, Graph, or Line objects. Access a specific object in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Count	Returns the number of report objects in the collection.	Read only
Item	Returns a report object. Depending on the type of item referenced, this can be a <i>BlobField Object</i> , <i>Page 620</i> , <i>FieldObject Object</i> , <i>Page 642</i> , <i>Text Object</i> , <i>Page 716</i> , <i>OLEObject Object</i> , <i>Page 661</i> , <i>CrossTabObject Object</i> , <i>Page 622</i> , <i>SubreportObject Object</i> , <i>Page 711</i> , <i>Box Object</i> , <i>Page 621</i> , <i>GraphObject Object</i> , <i>Page 651</i> , or a <i>LineObject Object</i> , <i>Page 660</i> . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)). The items in the collection are indexed in the order they were added to the report.	Read only
Parent	Reference to the Parent object (<i>Section Object</i> , <i>Page 700</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a report object directly, for example, ReportOptions(1).

ReportOptions Object

The **ReportOptions** Object provides properties for manipulating the options available in the report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
CaseInsensitiveSQLData	Returns/Sets Boolean value indicating if SQL data is case sensitive.	Read/Write
ConvertDateTimeToDate	Returns/Sets CRConvertDateTimeType (see table below) indicating what datetime fields are to be converted to (i.e., string, time, no conversion, etc.).	Read/Write

<i>Constant</i>	<i>Value</i>
crConvertDateTimeToDate	1
crConvertDateTimeToString	0
crKeepDateTimeType	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ConvertNullFieldToDefault	Returns/Sets Boolean value indicating if null database fields should be converted to the default value.	Read/Write
HasGroupTree	Returns/Sets Boolean value indicating if the preview window includes the Group Tree. You must also set the HasGroupTree property to TRUE in the <i>PrintWindowOptions Object</i> , <i>Page 683</i> , to have the Group Tree show in the preview window.	Read/Write
MorePrintEngineError-Messages	Returns/Sets Boolean value indicating if more print engine error messages are given at the report level. If they are on by default, you can turn this option off using this property.	Read/Write
Parent	Reference to the Parent object (<i>Report Object</i> , <i>Page 685</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
SaveDataWithReport	Returns/Sets Boolean value indicating if data is saved with the report.	Read/Write
SaveSummariesWithReport	Returns/Sets Boolean value indicating if summary values (i.e., group, report, cross-tab summaries) are to be saved with the report. You must also set SaveDataWithReport to TRUE.	Read/Write
TranslateDOSMemos	Returns/Sets Boolean value indicating if DOS Memos are translated.	Read/Write
TranslateDOSStrings	Returns/Sets Boolean value indicating if DOS Strings are translated.	Read/Write
UseIndexForSpeed	Returns/Sets Boolean value indicating if indexes are used.	Read/Write
VerifyOnEveryPrint	Returns/Sets Boolean value indicating if data is verified on every print.	Read/Write
ZoomMode	Returns/Sets CRZoomLevel (see table below) indicating what zoom level the preview should initially utilize when opened.	Read/Write

<i>Constant</i>	<i>Value</i>
crPageWidth	1
crWholePage	2

ReportSummaryInfo Object

The **ReportSummaryInfo** Object provides properties for retrieving information and setting options for summary information included in a report (i.e., author, title, etc.). A ReportSummaryInfo Object is obtained from the ReportSummaryInfo property of the *Report Object*, *Page 685*.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Author	Returns/Sets string specifying the name of the report author.	Read only
Comments	Returns/Sets string specifying the report comments.	Read/Write
Keywords	Returns/Sets string specifying the report keywords.	Read/Write
Name	Returns application name that created the report.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
Subject	Returns/Sets string specifying the report subject.	Read only
Template	Returns/Sets string specifying the report template.	Read/Write
Title	Returns/Sets string specifying the report title.	Read/Write

Section Object

Report areas contain at least one section. The **Section Object** includes properties for accessing information regarding a section of your report. This object holds on to a report object, then releases the report object when it is destroyed.

Properties

Property	Description	Read/Write
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Height	Returns/Sets the height of the section in twips.	Read/Write
Number	Returns the number associated with the section in the area (i.e., if the first section in an area, the number returned is 1).	Read only
Options	Returns <i>SectionOptions Object, Page 701</i> .	Read only
Parent	Reference to the Parent object (<i>Area Object, Page 615</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
ReportObjects	Returns <i>ReportObjects Collection, Page 696</i> . A heterogeneous collection of report objects.	Read only
Width	Returns the width of the section in twips.	Read only

SectionOptions Object

The **SectionOptions** Object represents a section found in a report. A section is a part of a report area which can be formatted separately from other sections in the report. This object provides properties for retrieving information and setting options for a section in your report (i.e., background color, new page after, etc.). A SectionOptions Object is obtained from the Options property of the *Section Object, Page 700*.

Each report area is comprised of sections which can be formatted independently of each other using the SectionOptions object. These section options are combined with the options set using the *AreaOptions Object, Page 617*, to make up the final appearance of a section and area of the report. The settings in the AreaOptions object will affect all sections within the area, while settings in the SectionOptions object will affect only the section they are set to.

If there is a conflict of settings between the AreaOptions object and the SectionOptions object, the object with an option set to TRUE will override the setting for the other object. For example, if the

AreaOptions object has the KeepTogether property set to TRUE, all sections within the area will have KeepTogether applied, even if the SectionOptions object for a section has KeepTogether set to FALSE. If, however, the AreaOptions object has KeepTogether set to FALSE, but a section within that area has KeepTogether set to TRUE, that section will have the KeepTogether format option applied.

If an area has only a single section, all options will be combined between both the AreaOption object and the SectionOptions object. All TRUE settings set in either object will result in a TRUE setting for the entire area and section. While changing format options for areas and sections in reports, be sure to keep track of settings in both the AreaOptions and SectionOptions objects.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
BackColor	Returns/Sets CRColor (see table below) indicating OLE color. You can use CRColor values or declare your own colors using OLE colors (i.e., using RGB function).	Read/Write

<i>Constant</i>	<i>Value</i>
crAqua	16776960
crBlack	0
crBlue	16711680
crFuchsia	16711935
crGray	8421504
crGreen	32768
crLime	65280
crMaroon	128
crNoColor	-1
crOlive	32896
crPurple	8388736
crRed	255
crSilver	12632256
crTeal	8421376
crWhite	16777215

Constant	Value
crYellow	65536

Property	Description	Read/Write
BackColorFormula	Returns/Sets string specifying a formula for conditionally setting the background color of a section.	Read/Write
FreeFormPlacement	Returns/Sets Boolean value indicating if section has free form placement.	Read/Write
KeepTogether	Returns/Sets Boolean indicating if section keeps together (doesn't break) across pages.	Read/Write
KeepTogetherFormula	Returns/Sets string specifying the formula for conditionally setting if a section should keep together (doesn't break) across pages.	Read/Write
NewPageAfter	Returns/Sets Boolean value indicating if there is a new page after the section is printed.	Read/Write
NewPageAfterFormula	Returns/Sets string specifying the formula for conditionally setting the report to start a new page after this section is printed.	Read/Write
NewPageBefore	Returns/Sets Boolean value indicating if there is a new page before the section is printed.	Read/Write
NewPageBeforeFormula	Returns/Sets string specifying the formula for conditionally setting the report to start a new page before this section is printed.	Read/Write
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
PrintAtBottomOfPage	Returns/Sets Boolean indicating if the section is printed at the bottom of the page	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
PrintAtBottomOf-PageFormula	Returns/Sets string specifying formula for conditionally printing section at the bottom of the page.	Read/Write
Report	Reference to <i>Report Object</i> , Page 685 .	Read only
ResetPageNumberAfter	Returns/Sets Boolean value indicating if the page number will be reset after the section is printed.	Read/Write
ResetPageNumber-AfterFormula	Returns/Sets string specifying the formula for conditionally resetting the page number.	Read/Write
SuppressIfBlank	Returns/Sets Boolean value indicating if the section is suppressed when containing no data.	Read/Write
SuppressIfBlankFormula	Returns/Sets string specifying the formula for conditionally suppressing a section when containing no data.	Read/Write
UnderlaySection	Returns/Sets Boolean value indicating if the section underlays the following sections.	Read/Write
UnderlaySectionFormula	Returns/Sets string specifying the formula for conditionally underlaying a section.	Read/Write
Visible	Returns/Sets Boolean value indicating whether a section is shown (visible).	Read/Write
VisibleFormula	Returns/Sets string specifying the formula for conditionally showing (making visible) a section.	Read/Write

Sections Collection

Sections can come from either the *Report Object*, [Page 685](#), or *Area Object*, [Page 615](#). The **Sections** Collection is a collection of section objects. Access a specific Section Object in the collection using the Item property.

- When from the Report Object, the sections object will contain all the sections in the report.
- When from the Area Object, the sections object will contain all the sections in the area only.

Properties

Property	Description	Read/Write
Application	Returns a reference to the <i>Application Object</i> , Page 611 , this object is associated with.	Read only
Count	Returns the number of section in the collection.	Read only
Item	Returns <i>Section Object</i> , Page 700 . Item has an index parameter that can be either a string reference to the area section (i.e., for areas with one section: "RH", "PH", "GHn", "D", "GFn", "PF", or "RF") or a numeric, 1-based index (i.e., Item (1) for the Report Header area. Numeric index for sections starts at 1 for first section in the area/report and continues in order of appearance. If the area has multiple sections, they are represented using a lowercase letter (i.e., "Da", "Db", etc.).	Read only
Parent	Reference to the Parent object (<i>Report Object</i> , Page 685 , or <i>Area Object</i> , Page 615).	Read only
Report	Reference to <i>Report Object</i> , Page 685 .	Read only

Remarks

Instead of using the Item property as shown, you can reference a section directly, for example, Sections ("Da").

SortField Object

The **SortField** Object includes properties for accessing information for record or group sort fields. It holds on to Report object, then releases the report object when destroyed. This object has an index instance variable to indicate its index.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Field	Returns/Sets <i>DatabaseFieldDefinition Object, Page 624</i> , which specifies the sort field used in the report.	Read/Write
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read/Write
SortDirection	Returns/Sets CRSSortDirection (see table below) indicating the direction the group/records are sorted in.	Read/Write

<i>Constant</i>	<i>Value</i>
crAscendingOrder	1
crDescendingOrder	0
crOriginalOrder	2
crSpecifiedOrder (Read only - this value cannot be set.)	3

SortFields Collection

The **SortFields** Collection is a collection of sort fields; can be either record sort field or group sort field. Access a specific SortField Object in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Count	Returns the number of sort fields in the collection.	Read only
Item	Returns <i>SortField Object, Page 706</i> . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)). The sort fields in the collection are indexed in the order they were added as sort fields the report.	Read only
Parent	Reference to the Parent object (<i>Report Object, Page 685</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a sort field directly, for example, SortFields(1).

Methods

Add

The **Add** method adds a record or group sort field.

Syntax

```
object.Add SortDirection, Field
```

Parameters

Parameter	Description
SortDirection	Specifies CRSortDirection (see table below) indicating the direction the field data should be sorted (i.e., ascending, descending, etc.).
Parameter	Description
Field	Specifies the field definition or field name.

Returns

SortField

Delete

The **Delete** method deletes a record or group sort field.

Syntax

```
object.Delete Index
```

Parameters

Parameter	Description
Index	Number indicating which sort field in the collection should be deleted. Numbering begins at 1 for the first sort field.

SpecialVarFieldDefinition Object

The **SpecialVarFieldDefinition** Object provides properties for retrieving information and setting options for a special field found in your report (i.e., last modification date, print date, etc.). A SpecialVarFieldDefinition Object is obtained from the *Field* property of the *FieldObject Object*, [Page 642](#).

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
Kind	Returns CRFieldKind (see table below) that specifies the “kind” of field (database, summary, formula, etc.).	Read only
<i>Constant</i>		<i>Value</i>
crDatabaseField		1
crFormulaField		2
crGroupNameField		5
crParameterField		6
crSpecialVarField		4
crSummaryField		3
crParameterField		6
<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the special var field.	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (<i>Report Object</i> , <i>Page 685</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only
SpecialVarType	Returns CRSpecialVarType (see table below) value indicating the type of special field (i.e., ReportTitle, PageNumber, etc.).	Read only
<i>Constant</i>		<i>Value</i>
crSVTDataDate		7
crSVTDataTime		8

<i>Constant</i>	<i>Value</i>
crSVTGroupNumber	6
crSVTModificationDate	2
crSVTModificationTime	3
crSVTPageNumber	5
crSVTPrintDate	0
crSVTPrintTime	1
crSVTRecordNumber	4
crSVTReportComments	11
crSVTReportTitle	10
crSVTTotalPageCount	9

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ValueType	Returns CRFieldValueType (see table below) which specifies the “type” of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateFormat	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7
crOleField	20
crPersistentMemoField	14

<i>Constant</i>	<i>Value</i>
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

SubreportObject Object

The **SubreportObject** Object represents a subreport found in a report. A subreport is a free-standing or linked report found within the main report. This object provides properties for retrieving information on the subreport (i.e., name, etc.).

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjectKind (see table below) that specifies what “kind” of object (i.e., box, cross-tab, field).	Read only

<i>Constant</i>	<i>Value</i>
crBlobFieldObject	9
crBoxObject	4
crCrossTabObject	8
crFieldObject	1
crGraphObject	7
crLineObject	3
crOleObject	6
crSubreportObject	5
crTextObject	2

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the subreport.	Read only
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

SummaryFieldDefinition Object

The **SummaryFieldDefinition** Object represents the summary used in a cross-tab, group, or report, or the summary used in a group or cross-tab graph.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
CrossTabObject	Returns <i>CrossTabObject Object, Page 622</i> , that contains the summary field. This property is for cross-tab summary fields only; if the summary field is not from a cross-tab you will get an exception.	Read only
FooterArea	Returns <i>Area Object, Page 615</i> , indicating the Group/Report Footer area of the summary. This property is for group or report summary fields only; if the summary is from a cross-tab you will get an exception using this property.	Read only
ForCrossTab	Returns Boolean value indicating if summary is used in a cross-tab.	Read only

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
HeaderArea	Returns <i>Area Object</i> , <i>Page 615</i> , indicating the Group/Report Header area of the summary. If the summary is from a cross-tab you will get an exception using this property.	Read only
Kind	Returns CRFieldKind which specifies what “kind” of field (database, summary, formula, etc.).	Read only
<i>Constant</i>		<i>Value</i>
crDatabaseField		1
crFormulaField		2
crGroupNameField		5
crParameterField		6
crSpecialVarField		4
crSummaryField		3
<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Name	Returns the name of the summary field for a group or report summary field. Cross-tab summaries do not have a unique name so an empty string will be returned.	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (<i>Report Object</i> , <i>Page 685</i>).	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only
SummaryType	Returns value of type CRSummaryType (see table below) indicating the type of summary (i.e., sum, average, etc.).	Read only

<i>Constant</i>	<i>Value</i>
crSTAverage	1
crSTCount	6
crSTDistinctCount	9
crSTMaximum	4
crSTMinimum	5
crSTPopStandardDeviation	8
crSTPopVariance	7
crSTSampleStandard- Deviation	3
crSTSampleVariance	2
crSTSum	0

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
ValueType	Returns CRFieldValueType (see table below) which specifies the "type" of value found in the field.	Read only

<i>Constant</i>	<i>Value</i>
crBitmapField	17
crBlobField	15
crBooleanField	9
crChartField	21
crCurrencyField	8
crDateField	10
crDateTimeField	16
crIconField	18
crInt16sField	3
crInt16uField	4
crInt32sField	5
crInt32uField	6
crInt8sField	1
crInt8uField	2
crNumberField	7
crOleField	20
crPersistentMemoField	14

<i>Constant</i>	<i>Value</i>
crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

SummaryFieldDefinitions Collection

The **SummaryFieldDefinitions** Collection is a collection of summary field definitions. Can be from either *CrossTabObject Object, Page 622*, or *Report Object, Page 685*. Access a specific SummaryFieldDefinition Object in the collection using the Item property.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Count	Returns the number of summary fields in the collection.	Read only
Item	Returns <i>SummaryFieldDefinition Object, Page 712</i> . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1)). The items in the collection are indexed in the order they were added to the report.	Read only
Parent	Reference to the Parent object <i>CrossTabObject Object, Page 622</i> , or <i>Report Object, Page 685</i> .	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

Remarks

Instead of using the Item property as shown, you can reference a summary field directly, for example, SummaryFieldDefinitions (1).

Text Object

The Text Object represents a text object found in a report. This object provides properties for retrieving information and setting options for a text object in your report.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
Kind	Returns CRObjectKind (see table below) that specifies what “kind” of object (i.e., box, cross-tab, field).	Read only
<i>Constant</i>		<i>Value</i>
crBlobFieldObject		9
crBoxObject		4
crCrossTabObject		8
crFieldObject		1
crGraphObject		7
crLineObject		3
crOLEObject		6
crSubreportObject		5
crTextObject		2
<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object (<i>Section Object, Page 700</i>).	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

TrackCursorInfo Object

The **TrackCursorInfo** Object provides properties/methods for tracking the position of the cursor in the report and for setting cursor icons for the specified areas of the report. Cursor icon availability will vary depending on your development platform.

Properties

<i>Property</i>	<i>Description</i>	Read/Write
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
DetailAreaCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor to be displayed when the cursor is moved over the Details Area. Check your operating system documentation for a list of cursors available on your system. If you choose an unavailable cursor, the default (arrow) cursor will be used.	Read/Write
DetailAreaFieldCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor to be displayed when the cursor is moved over fields in the Details Area. Check your operating system documentation for a list of cursors available on your system. If you choose an unavailable cursor, the default (arrow) cursor will be used.	Read/Write

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
GraphCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor to be displayed when the cursor is moved over a graph. Check your operating system documentation for a list of cursors available on your system. If you choose an unavailable cursor, the default (arrow) cursor will be used.	Read/Write
GroupAreaCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor to be displayed when the cursor is moved over the Group Area. Check your operating system documentation for a list of cursors available on your system. If you choose an unavailable cursor, the default (arrow) cursor will be used.	Read/Write
GroupAreaFieldCursor	Returns/Sets CRTrackCursor (see table below) indicating what type of cursor to be displayed when the cursor is moved over fields in the Group Area. Check your operating system documentation for a list of cursors available on your system. If you choose an unavailable cursor, the default (arrow) cursor will be used.	Read/Write

<i>Constant</i>	<i>Value</i>
crAppStartingCursor	12
crArrowCursor	1
crCrossCursor	2
crDefaultCursor	0
crHelpCursor	13
crIBeamCursor	3
crIconCursor	15
crMagnifyCursor	99
crNoCursor	10
crSizeAllCursor	5
crSizeCursor	14
crSizeNESWCursor	7
crSizeNSCursor	9
crSizeNWSECursor	6
crSizeWECursor	8
crUpArrowCursor	4
crWaitCursor	11

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Parent	Reference to the Parent object <i>PrintWindowOptions Object, Page 683</i> .	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only

View Object

The **View** Object is returned when previewing a report. A View represents a single view on the report. A View always lives inside a window. While there are two types of views available in Seagate Crystal Reports (Design and Preview), the Crystal Report Engine supports only the Preview window view. This object holds on to the *Report Object, Page 685*, then releases it when it is destroyed.

A View object is created when a report is first generated and displayed in the Preview window. If a user drills-down on summary values or on graph data, additional View objects are created as additional drill-down views appear in the Preview window. Each Preview tab that appears in the Preview window represents a separate View object. Only the View object for the currently active Preview tab can be manipulated at runtime.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , <i>Page 611</i> , this object is associated with.	Read only
DisplayedPageNumber	Returns the page number currently displayed in the current view. You will receive an incorrect result if you try to obtain this value immediately after using a method such as <i>ShowNextPage</i> , <i>Page 722</i> , that causes a change to the preview window. Call the Visual Basic function, DoEvents (see Visual Basic documentation for more information) before requesting the page number display.	Read only
Parent	Reference to the Parent object <i>Window Object</i> , <i>Page 724</i> .	Read only
Report	Reference to <i>Report Object</i> , <i>Page 685</i> .	Read only

Methods

Close

The **Close** method closes the main view. Because only one view is supported, this method will apply to the main view (preview) only. The Preview window will be closed.

Syntax

```
object.Close
```

Export

The **Export** method exports the main view to selected format and destination; prompts for options. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.Export
```

NextMagnification

The **NextMagnification** method shows the view at the “next magnification” in the series (i.e., page width - full page - 100%). Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.NextMagnification
```

PrintOut

The **PrintOut** method sends the view to printer specified in the report to be printed. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.PrintOut
```

ShowFirstPage

The **ShowFirstPage** method shows the first page of the view. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ShowFirstPage
```

ShowLastPage

The **ShowLastPage** method shows the last page of the view. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ShowLastPage
```

ShowNextPage

The **ShowNextPage** method shows the next page of the view. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ShowNextPage
```

ShowNthPage

The **ShowNthPage** method shows the nth page of the view. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ShowNthPage (PageNumber)
```

Parameters

Parameter	Description
PageNumber	Specifies the number of the page to be shown.

ShowPreviousPage

The **ShowPreviousPage** method shows the previous page of the view. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ShowPreviousPage
```

ZoomPreviewWindow

The **ZoomPreviewWindow** method zooms the view in and out, showing the report as a percentage of the full size of the report page. Because only one view is supported, this method will apply to the main view (preview) only.

Syntax

```
object.ZoomPreviewWindow (ZoomLevel)
```

Parameters

Parameter	Description
ZoomLevel	Specifies the percentage of magnification for the view (i.e., 50%, 75%, etc.).

Views Collection

The **Views** Collection is a collection of views for a report. A *View Object*, [Page 719](#), is created when a report is first generated and displayed in the Preview window. If a user drills-down on summary values or on graph data, additional View objects are created as additional drill-down views appear in the Preview window. Each Preview tab that appears in the Preview window represents a separate View object. Only the View object for the currently active Preview tab can be manipulated at runtime.

The Views collection contains all View objects for the report. However, since only the currently active view can be accessed, the Count property will always contain a value of 1. The currently active View object can be accessed by passing 1 to the Item property. For example:

```
Set activeView = viewsCollection.Item(1)
```

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object</i> , Page 611 , this object is associated with.	Read only
Count	Returns the number of views in the collection. Because only one view is supported (preview), this number should always be 1.	Read only
Item	Returns <i>View Object</i> , Page 719 . Item has an index parameter that is a numeric, 1-based index (i.e., Item (1) returns the main view). Because only a single View object can be accessed at any time, the currently active view, any index other than 1 will return an error.	Read only
Parent	Reference to the Parent object <i>Window Object</i> , Page 724 .	Read only
Report	Reference to <i>Report Object</i> , Page 685 .	Read only

Remarks

Instead of using the Item property as shown, you can reference a view directly, for example, Views(1).

Window Object

The **Window Object** is obtained through View.Parent. It is the window that previews the report. A Window represents the highest-level window associated with a single report. In the Crystal Report Engine and the Crystal Report Automation Server, it is a preview window.

Properties

<i>Property</i>	<i>Description</i>	<i>Read/Write</i>
Application	Returns a reference to the <i>Application Object, Page 611</i> , this object is associated with.	Read only
ControlsVisible	Returns/Sets Boolean value indicating if window controls are visible in the preview window.	Read/Write
Parent	Reference to the Parent object <i>Report Object, Page 685</i> .	Read only
Report	Reference to <i>Report Object, Page 685</i> .	Read only
Views	Returns <i>Views Collection, Page 723</i> , a collection of views in the window.	Read only
WindowHandle	Returns the handle of the window.	Read only

Methods

Close

The **Close** method closes the window, and consequently all views in the window.

Syntax

```
object.Close
```

Events

ActivatePrintWindow

The **ActivatePrintWindow** event occurs when the print window is activated (i.e., received focus from another window).

Syntax

```
Event ActivatePrintWindow ()
```

Remarks

This event is enabled using the `ActivatePrintWindowEventEnabled` property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the `EventInfo` property of the *Report Object*, *Page 685*.

*NOTE: For more information on events, see *Object Library Events*, *Page 611*.*

CancelButtonClicked

The **CancelButtonClicked** event occurs when the Cancel button is clicked.

Syntax

```
Event CancelButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign <code>useDefault = FALSE</code> if not using default behavior defined in CPEAUT (i.e., if you do not want the cancel button to cancel moving to the last page or processing the report).

Remarks

Before enabling this event, the `HasCancelButton` property of the *PrintWindowOptions Object*, *Page 683*, must be set to TRUE. This event is enabled using the `PrintWindowButtonEnabled` property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the `EventInfo` property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see Object Library Events, Page 611.

CloseButtonClicked

The **CloseButtonClicked** event occurs when the Close button is clicked either to close a view or the entire preview window itself.

Syntax

```
Event CloseButtonClicked (ViewIndex As Integer,  
useDefault As Boolean)
```

Parameters

Parameter	Description
ViewIndex	Specifies the index of the view that is to be closed.
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the view/preview window to close).

Remarks

Before enabling this event, the HasCloseButton property of the *PrintWindowOptions Object*, Page 683, must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, Page 636, obtained through the EventInfo property of the *Report Object*, Page 685.

NOTE: For more information on events, see Object Library Events, Page 611.

ClosePrintWindow

The **ClosePrintWindow** event occurs when the print window is closed.

Syntax

```
Event ClosePrintWindow (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the print window to close).

Remarks

This event is enabled using the ClosePrintWindowEventEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see Object Library Events, Page 611.

DeactivatePrintWindow

The **DeactivatePrintWindow** event occurs when the print window is deactivated (i.e., when any other window takes the focus from the print window).

Syntax

```
Event DeactivatePrintWindow ()
```

Remarks

This event is enabled using the ActivatePrintWindowEventEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see Object Library Events, Page 611.

DrillOnDetail

The **DrillOnDetail** event occurs when you drill down on a field in the Details area of the report.

Syntax

```
Event DrillOnDetail (FieldValues, SelectedFieldIndex As Long,  
useDefault As Boolean)
```

Parameters

Parameter	Description
FieldValues	Specifies an array of FieldValue objects (see <i>FieldValue Object (32-bit only)</i> , <i>Page 644</i>). Each object contains information about a single field for the Detail record that was drilled down on. FieldValues may contain database field values, formula field values and summary field values, and group name field values of the group the record belongs to.
SelectedFieldIndex	Specifies the selected field index in the array that points to the actual FieldValue. If no field is selected, 0 is returned.

<i>Parameter</i>	<i>Description</i>
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want drill down to occur).
ReportName	Reserved for future use.

Remarks

Before enabling this event, the CanDrillDown property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE.

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

DrillOnGroup

The **DrillOnGroup** event occurs when drilling down (double-clicking) on a group field in the preview window.

Syntax

```
Event DrillOnGroup (GroupNameList, DrillType As CRDrillType,  
useDefault As Boolean)
```

Parameters

<i>Parameter</i>	<i>Description</i>
GroupNameList	Specifies an array indicating the path to the group that was drilled down on (separated by "/").
DrillType	Specifies CRDrillType (see table below) indicating what type of drill down occurred.

<i>Constant</i>	<i>Value</i>
crDrillOnGraph	2
crDrillOnGroup	0
crDrillOnGroupTree	1

<i>Parameter</i>	<i>Description</i>
useDefault	Specifies Boolean value indicating whether the default settings be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want drill down to occur).
ReportName	Reserved for future use.

Remarks

Before enabling this event, the CanDrillDown property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE. This event is enabled using the GroupEventEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

ExportButtonClicked

The **ExportButtonClicked** event occurs when the Export button is clicked.

Syntax

```
Event ExportButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the Export dialog box to appear).

Remarks

Before enabling this event, the HasExportButton property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

FirstPageButtonClicked

The **FirstPageButtonClicked** event occurs when the button which navigates through the report to the first page is clicked.

Syntax

```
Event FirstPageButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want to go to the first page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

GroupTreeButtonClicked

The **GroupTreeButtonClicked** event occurs when the Group Tree button is clicked to show/hide the Group Tree in the preview window.

Syntax

```
Event GroupTreeButtonClicked (Visible As Boolean)
```

Parameters

Parameter	Description
Visible	Specifies Boolean value indicating whether the Group Tree is visible/hidden after clicking the button.

Remarks

Before enabling this event, the HasGroupTree property of the *PrintWindowOptions Object*, [Page 683](#), and the HasGroupTree property of the *ReportOptions Object*, [Page 697](#) must be set to TRUE. This event is enabled using the GroupEventEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

LastPageButtonClicked

The **LastPageButtonClicked** event occurs when the button which navigates through the report to the last page is clicked.

Syntax

```
Event LastPageButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want to go to the last page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object*, *Page 683*, must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see *Object Library Events*, *Page 611*.

NextPageButtonClicked

The **NextPageButtonClicked** event occurs when the button which navigates through the report to the next page is clicked.

Syntax

```
Event NextPageButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default values should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want to go to the next page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object*, *Page 683*, must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see *Object Library Events*, *Page 611*.

PrevPageButtonClicked

The **PrevPageButtonClicked** event occurs when the button which navigates through the report to the previous page is clicked.

Syntax

```
Event PrevPageButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default values should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want to go to the previous page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object*, *Page 683*, must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see *Object Library Events*, *Page 611*.

PrintButtonClicked

The **PrintButtonClicked** event occurs when the Print button is clicked.

Syntax

```
Event PrintButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the Print dialog box to appear).

Remarks

Before enabling this event, the HasPrintButton property of the *PrintWindowOptions Object*, *Page 683*, must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, *Page 636*, obtained through the EventInfo property of the *Report Object*, *Page 685*.

NOTE: For more information on events, see Object Library Events, Page 611.

PrintSetupButtonClicked

The **PrintSetupButtonClicked** event occurs when the Print Setup button is clicked.

Syntax

```
Event PrintSetupButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default setting should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the Printer Setup dialog box to appear).

Remarks

Before enabling this event, the **HasPrintSetupButton** property of the *PrintWindowOptions Object*, Page 683, must be set to TRUE. This event is enabled using the **PrintWindowButtonEnabled** property of the *EventInfo Object (32-bit only)*, Page 636, obtained through the **EventInfo** property of the *Report Object*, Page 685.

NOTE: For more information on events, see Object Library Events, Page 611.

RefreshButtonClicked

The **RefreshButtonClicked** event occurs when the Refresh button is clicked.

Syntax

```
Event RefreshButtonClicked (useDefault As Boolean)
```

Parameters

Parameter	Description
useDefault	Specifies Boolean value indicating whether the default setting should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the report to refresh).

Remarks

Before enabling this event, the HasRefreshButton property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

SearchButtonClicked

The **SearchButtonClicked** event occurs when the Search button is clicked.

Syntax

```
Event SearchButtonClicked (SearchString As String,  
useDefault As Boolean)
```

Parameters

Parameter	Description
SearchString	Specifies the string entered by the user that will be searched for.
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the search for the string to occur).

Remarks

Before enabling this event, the HasSearchButton property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE. This event is enabled using the PrintWindowButtonEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

ShowGroup

The **ShowGroup** event occurs when you click a group in the Group Tree.

Syntax

```
Event ShowGroup (GroupNameList, useDefault As Boolean)
```

Parameters

Parameter	Description
GroupNameList	Specifies an array indicating the path to the group that was drilled down on (separated by "/").
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (i.e., if you do not want the group the user clicked on to be shown in the preview window).
ReportName	Reserved for future use.

Remarks

Before enabling this event, the HasGroupTree property of the *PrintWindowOptions Object*, [Page 683](#), and the HasGroupTree property of the *ReportOptions Object*, [Page 697](#) must be set to TRUE. This event is enabled using the GroupEventEnabled property of the *EventInfo Object (32-bit only)*, [Page 636](#), obtained through the EventInfo property of the *Report Object*, [Page 685](#).

NOTE: For more information on events, see *Object Library Events*, [Page 611](#).

ZoomLevelChanging

The **ZoomLevelChanging** event occurs when the zoom level of the report preview is changed.

Syntax

```
Event FirstPageButtonClicked (ZoomLevel As Integer)
```

Parameters

Parameter	Description
ZoomLevel	Specifies integer indicating the percentage of the magnification for viewing the report. Can be CRZoomLevel (see table below) for Page Width and Whole Page. Otherwise, indicate zoom level percentage as an integer.

Constant	Value
crPageWidth	1
crWholePage	2

Remarks

Before enabling this event, the HasZoomControl property of the *PrintWindowOptions Object*, [Page 683](#), must be set to TRUE.

NOTE: For more information on events, see *Object Library Events, Page 611*.

Error Codes

The Crystal Report Engine Automation Server generates the following error codes when necessary:

<i>Value</i>	<i>Error Code</i>	<i>Description</i>
30001	CPEAUT_STANDARD_OUTOFMEMORY	There is not enough memory to complete the action.
30002	CPEAUT_STANDARD_INVALIDHANDLE	A handle has been specified that is invalid or does not exist.
30003	CPEAUT_STANDARD_INVALIDPOINTER	A pointer has been specified that is invalid or does not exist.
30004	CPEAUT_STANDARD_BADINDEX	An invalid (bad) value has been specified for an index.
30005	CPEAUT_STANDARD_TYPEMISMATCH	Wrong data type was used.
30006	CPEAUT_STANDARD_UNKNOWNNAME	The specified name can not be found in the report.
30007	CPEAUT_STANDARD_MEDIUMFULL	The storage medium (disk drive) is full.
30008	CPEAUT_STANDARD_NOTIMPL	Internal error.
30009	CPEAUT_ERR_NOTGROUPAREA	A group area was expected, but the area specified was not a group area.
30010	CPEAUT_ERR_STRINGOUTOFRANGE	The string value passed is out of the range of possible values.
30011	CPEAUT_ERR_NODATABASE	There is no database available.
30012	CPEAUT_ERR_INVALIDENUMVALUE	An invalid enumerated value has been set.
30013	CPEAUT_ERR_NOTDETAILAREA	A details area was expected, but the area specified was not a details area.
30014	CPEAUT_ERR_NOTAREASECTIONCOLLECTION	The object specified is not an Areas or Sections collection.
30015	CPEAUT_ERR_32NOTSUPPORTED	A 32-bit only feature has been evoked in an environment that does not support it.
30016	CPEAUT_ERR_INVALIDKEY	A key has been specified that is invalid or does not exist.

<i>Value</i>	<i>Error Code</i>	<i>Description</i>
30017	CPEAUT_ERR_FEATURENOTENABLED	A feature has been evoked that is not yet enabled.
30018	CPEAUT_ERR_DATABASELINKEXIST	An error has occurred due to the database links.
30019	CPEAUT_ERR_CREATETEMPFILEFAILED	Creation of a temporary file has been unsuccessful.
30020	CPEAUT_ERR_RENDERTOHTMLFAILED	Rendering to using <i>RenderHTML</i> , Page 663 HTML has been unsuccessful.

Report Engine Error Codes

The Crystal Report Engine Automation Server will also report error codes passed to it from the Crystal Report Engine API. The values of these codes are identical to the Report Engine error code value plus 20000. For example the value of the Report Engine error PE_ERR_NOTENOUGHMEMORY is 500. The value of the equivalent error reported by the automation server is 20500.

For complete information on these errors, search for *Report Engine error codes* in the Developer's online Help.

10

Visual Component Library Reference

What you will find in this chapter...

TCrpe Component, Page 740

TCrpePrinterSetup Class, Page 740

PROPERTIES, Page 740

METHODS, Page 824

EVENTS, Page 836

TUTORIALS, Page 837

TCrpe Component

UCrpe Unit

The UCrpe unit contains the declaration for the TCrpe component and its associated objects.

When you add a component declared in this unit to a form, the unit is automatically added to the “uses” clause of that form’s unit.

TCrpePrinterSetup Class

Unit

UCrpe

Description

Design time; Runtime

By default this is set to False.

This object is used to control whether or not the TCrpe object is to request printer setup information at Runtime. There is no need to set PrinterName, PrinterDriver, or PrinterPort when this property is enabled.

TCrpePrinterSetup Properties

There are two Properties supported by the TCrpePrinterSetup object: UserPrinterSetupEnabled Property, and *ReportPrinterSetup*, *Page 800*.

PROPERTIES

Action

Description

Triggers the printing of the report.

Usage

```
[form.]Report.Action:=1;
```

For example:

```
Crpe1.Action := 1;  
«Prints the specified report.»
```

Remarks

Set the Action property to 1 in your procedure code (Crpe1.Action := 1) to print the report in response to a user event.

Data Type

Byte

Availability

Runtime

Connect

Description

Logs on to an SQL server.

Usage

```
[form.]Report.Connect[ := Name; UserID; Password; DatabaseQualifier];
```

For example:

```
Crpe1.Connect := 'DSN=Accounting; UID=734; PWD=bigboard;  
DSQ=Administration';
```

«Connects to the “Administration” database on the “Accounting” server using the user ID #734 and the password “bigboard”.»

Remarks

- Enter the parameters necessary to log on to the SQL server that you need to be activated for your report.
- Parameters should be in the following format:

```
DSN = name;UID = userID;PWD = password;DSQ = database qualifier  
— name is the server name,
```

- *user ID* is the name you have been assigned for logging on to the SQL server,
- *password* is the password you have been assigned for logging on to the SQL server, and
- *database qualifier* is the database name if your server uses the database concept.

NOTE: Before you can use this property, you must install the ODBC or native SQL driver for whatever SQL database you are planning to use, and put the Database\BIN location in your path.

NOTE: This parameter is required only when it is applicable to the ODBC or native SQL driver you are using.

Data Type

String

Availability

Design time; Runtime

CopiesToPrinter

Description

Specifies the number of copies to be printed if you are printing to a printer.

Usage

```
[form.]Report.CopiesToPrinter[ :=NumCopies];
```

For example:

```
Crp1.CopiesToPrinter := 3;
```

```
«Prints three copies of the specified report.»
```

Remarks

The number you enter must not be a zero or a negative value.

Data Type

Integer

Availability

Design time; Runtime

DataFiles

Description

Specifies the location of the database files or tables used in the report.

Usage

```
[ form. ]Report.DataFiles[ArrayIndex][ :=Location];
```

- Enter the file name and path of each database file or table in your report for which you want to change the location.
- Use a separate line of code for each file for which you want to change the location.
- The order of files in the array must conform to the order of files in the report. (You can use the Database | Set Location command to determine the order of files in the report.)
- The first file in the report is array index [0], the second file is [1], etc.

For example, to change the location of the first and third files in a report (first.dbf and third.dbf) to the C:\NEW directory, use the following syntax:

```
Crpel.DataFiles[0] := 'c:\new\first.dbf';
Crpel.DataFiles[2] := 'c:\new\third.dbf';
```

Remarks

- DataFiles is an array property that is available at runtime only.
- Use this property if you want to run the report with files in different locations than specified in the report.
- When using this property, you do not have to change the locations of all files in the report. Just make certain that the array index for each file you do change matches the position of that file in the report.
- This property is cleared once the print job is finished. If you print a second time, the program reverts to the locations as originally specified in the report.

NOTE: The DataFilesLocation property overrides any changes you make using this property.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

Access Notes Sparse Array Properties, Page 837

DataFilesLocation

Description

This property is used to set the directory location of the data files at runtime.

Use this property when the location of the data files is different than the report and that all the tables are in the same location.

Usage

```
[form.]Report.DataFilesLocation[ :=Location];
```

For example:

```
Crpe1.DataFilesLocation := 'C:\Account' ;
```

«Sets the database location for the whole report.»

Remarks

- Using this property overrides any changes you make using the *DataFiles, Page 743*.
- This property requires that the table name be the same as development.

Data Type

String

Availability

Runtime

Destination

Description

Specifies the destination to which your report is to be printed (Window, Printer, File or E-mail).

Usage

```
[form.]Report.Destination[ :=TDestination];
```

For example:

```
Crpel.Destination := toWindow;
```

«Sends the specified report to a preview window.»

Remarks

Select one of the following print destinations:

- **toWindow**

Sends the report to a preview window.

- **toPrinter**

Sends the report to a printer.

- **toFile**

Prints the report to a disk file for printing at a later time or for importing into other applications. If you select this property, you will also have to set the *PrintFileName*, *Page 786*, and *PrintFileType*, *Page 791*.

- **toEMailViaMAPI**

Sends the report to another person on your network via MAPI e-mail (Microsoft Mail). The report is attached to the e-mail letter in the format specified by the *PrintFileType*, *Page 791*.

- **toEMailViaVIM**

Sends the report to another person on your network via VIM e-mail (cc:Mail). The report is attached to the e-mail letter in the format specified by the *PrintFileType*, *Page 791*.

Data Type

A value indicating the print destination. See Remarks.

Availability

Design time; Runtime

DetailCopies

Description

Specifies the number of copies of each record in the Details section that the program is to print.

Usage

```
[form.]Report.DetailCopies[ := NumCopies];
```

For example:

```
Crpe1.DetailCopies := 3;
```

«Specifies that three copies of each record in the details section are to be printed.»

Remarks

If DetailCopies is set to a value less than or equal to zero, the value is ignored and one copy of the Detail section of the report is printed.

Data Type

Integer

Availability

Design Time; Runtime

DialogParentHandle

Description

Specifies the handle of the parent window. The program uses this handle to determine the window within which it centers any dialog boxes it displays (progress dialog boxes, parameter field prompt dialog boxes, and so forth).

Usage

```
[form.]Report.DialogParentHandle[ := HWND];
```

For example:

```
Crpe1.DialogParentHandle:= ParentHwnd;
```

«Specifies the handle of the parent for all dialog boxes the custom control will display.»

Remarks

Does not affect the placement of the preview window. Preview window placement is determined by the WindowLeft, WindowTop, WindowHeight, and WindowWidth properties.

Data Type

hWnd

Availability

Runtime only

DiscardSavedData

Description

If data is saved with the specified report, setting this property to True discards the data.

Usage

```
[form.]Report.DiscardSavedData[ :={True|False}];
```

For example:

```
Crp1.DiscardSavedData := True;
```

«Discards the data saved with the specified report.»

Data Type

Boolean

Availability

Design time; Runtime

EMailCCLList

Description

Specifies the “CC” list to which you want your e-mail message sent.

Usage

```
[form.]Report.EMailCCLList[ :=CCLList];
```

For example:

```
Crpel.EMailCCList := 'John Brown; Jane Doe';  
«Sends a CC of the e-mail message to both John Brown and Jane Doe.»
```

Remarks

- Applies to both VIM and MAPI.
- Multiple names must be separated by a semicolon.

Data Type

String

Availability

Design time; Runtime

EMailMessage

Description

Specifies the string you want to appear as the body of your e-mail message.

Usage

```
[form.]Report.EMailMessage[ :=Message ];
```

For example:

```
Crpel.EMailMessage := 'The meeting is at 4:00';  
«Sets "The meeting is at 4:00" as the body of the e-mail message.»
```

Remarks

Applies to both MAPI and VIM.

Data Type

String

Availability

Design time; Runtime

EMailSubject

Description

Specifies the subject line in your e-mail message.

Usage

```
[form.]Report.EMailSubject[:=Subject];
```

For example:

```
Crpel.EMailSubject := 'Staff meeting';
```

«Sets "Staff meeting" as the subject line in an e-mail message.»

Remarks

Applies to both MAPI and VIM.

Data Type

String

Availability

Design time; Runtime

EMailToList

Description

Specifies the "To" list to which you want your e-mail message directed.

Usage

```
[form.]Report.EMailToList[:=ToList];
```

For example:

```
Crpel.EMailToList := 'Jane Doe';
```

«Makes "Jane Doe" the only name in the "To" list.»

Remarks

- Applies to both MAPI and VIM.
- Multiple names must be separated by a semicolon.

Data Type

String

Availability

Design time; Runtime

EMailVIMBCCList

Description

Specifies the “Blind CC” list to which you want your e-mail message copied.

Usage

```
[form.]Report.EMailVIMBCCList[ :=BCCList];
```

For example:

```
Crpel.EMailVIMBCCList := 'John Jacobs;Jane Doe';  
«Makes “John Jacobs” and “Jane Doe” the names for the BCC list.»
```

Remarks

- Applies to VIM only (cc:Mail), not MAPI.
- Multiple names must be separated by a semicolon.

Data Type

String

Availability

Design time; Runtime

UserPrinterSetup.Enabled

Description

This property is used to enable or disable the UserPrinterSetup Object.

Usage

```
[form.]Report.UserPrinterSetup.Enabled[:={True|False}];
```

For example:

```
Crpel.UserPrinterSetup.Enabled:=True;
```

«Allows the printer settings to be prompted for by the printer dialog box.»

Remarks

Will prompt using a printer dialog box when the execute command is issued.

Data Type

Boolean

Availability

Design time; Runtime

ExchangeFolder

Description

Specifies the Exchange path to export a file, when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangeFolder[:={string}];
```

For example:

```
Crpel.ExchangeFolder :=  
"c:\Microsoft\Exchange\NewRpt.rpt";
```

«Send the report to the file, NEWRPT.RPT in the subdirectory \Microsoft\Exchange.»

Data Type

String

Availability

Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.DestinationOptions set to a UXDExchangeOptions structure.

ExchangePassword

Description

Specifies the Exchange password when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangePassword[:={string}];
```

For example:

```
Crpe1.ExchangePassword := "pickle";
```

```
«The Exchange password is pickle.»
```

Data Type

String

Availability

Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.DestinationOptions set to a UXDExchangeOptions structure.

ExchangeProfile

Description

Specifies the Exchange Profile when you want to export to Microsoft Exchange.

Usage

```
[form.]Report.ExchangeProfile[ := {string} ];
```

For example:

```
Crpel.ExchangeProfile :=  
"James Anderson";
```

«Specifies James Anderson as the Exchange Profile.»

Remarks

Usually your profile is your name.

Data Type

String

Availability

Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.DestinationOptions set to a UXDExchangeOptions structure

Execute

Description

Triggers the printing of a report.

Usage

```
[form.]Report.Execute;
```

For example:

```
Crpel.Execute;
```

Remarks

- Triggers other events before starting the report.
- Execute should be used in place of the *Action*, *Page 740*.

Data Type

Boolean

Availability

Runtime

Formulas

Description

Specifies a new string for an existing formula.

Usage

```
[form.]Report.Formulas(ArrayIndex) [ :='FormulaName=FormulaText' ];
```

Enter the formula name and the string that you want to replace the existing string for each formula that you want to change in your report.

For example, to change a formula @COMMISSION to {file.SALES}*1, and a second formula @TOTAL to {file.SALES} + {file.COMMISSION}, enter the following:

```
Crpel.Formulas[0] := 'COMMISSION={file.SALES} * .1';
Crpel.Formulas[1] := 'TOTAL={file.SALES} + {file.COMMISSION}' ;
```

Remarks

- Formulas is an array property that is available at runtime only.
- Use a separate line of code for each formula you want to change.
- Change only those formulas that you want to change.
- The first formula you change must be assigned array index [0], the second must be assigned array index [1], etc.

- The new formula string must conform to Seagate Crystal Reports syntax requirements.
- This property is cleared once the print job is printed. If you print a second time, the program reverts to the formulas as originally specified in the report.

NOTE: Spaces are significant in formula names. For this reason, the equal sign must follow the formula name with no intervening spaces.

NOTE: The @ sign is not used when designating a formula name in this property. You can't use this property to create new formulas. You can only use it to change existing formulas.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

GraphData

Description

Sets the data used for a specified graph.

Usage

```
[form. ]Report.GraphData(ArrayIndex%)
[= sectionCode; graphNum; row; col; field; direction];
```

For example:

```
[form. ]Report.GraphData(ArrayIndex%)
[= sectionCode; graphNum; row; col; field; direction];
```

Remarks

With GraphData, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action=1” call. The array index value for GraphData simply specifies the sequence number for the change. Thus:

```
CrystalReport1.GraphData(0) = "GH1; 3; Group1;
Group2; 666; COLANDROW"
```

when making changes to one graph only, but

```
CrystalReport1.GraphData(0) = "HEADER; 3; Group1;
Group2; 666; COLANDROW"
CrystalReport1.GraphData(1) = "GH1; 3; Group1;
Group2; 666; COLANDROW"
```

when making changes to more than one graph.

Use the following table as a guide in supplying the required values for this property:

Parameter	Description	Expected value
sectionCode	Specifies the section in which you want to modify a graph.	Please refer to the section code table <i>Section Codes (16-bit)</i> , <i>Page 495</i> , or <i>Section Codes (32-bit)</i> , <i>Page 494</i> .
graphNum	The number of the graph within the section you want to modify.	Graphs in a section are numbered, starting with zero, left to right first, then top to bottom.
row	The Group number in the report used to create rows in the graph.	GROUP1, GROUP2, GROUP3,..., GROUP9
col	The Group number in the report used to create columns in the graph.	GROUP1, GROUP2, GROUP3,..., GROUP9
field	The summarization field containing values to be used as the value of each riser in the graph.	The first summarization field added to a report is numbered 0, the second added to a report is numbered 1, etc.
direction	Whether the values in the rows, the columns, or both are used to create the graph.	ROWS, COLS, ROWANDCOL, or COLANDROW

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetGraphData, Page 289

GraphOptions

Description

Sets a number of options for the specified graph.

Usage

```
[form.]Report.GraphOptions[ArrayIndex]
[ :='sectionCode; graphNum; fontFace; barDirection; labelRisers;
gridLines; legend; max; min'];
```

For example:

```
Crpel.GraphOptions[0] := 'Title;0;Arial;V;T;F;F;10;5';
```

«Sets the font to Arial, sets vertical bars, shows a data value on every riser (labelRisers = T), and toggles the grid lines off in the first Graph in the Title section, turns Legend off with a maximum of 10 and a minimum of 5 on the Y axis.»

Remarks

- With GraphOptions, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the *Action, Page 740*, equal to 1. The array index value for GraphOptions simply specifies the sequence number for the change. Thus:

```
Crpel.GraphOptions[0] := 'Title; 0; Arial; H; T; F; legend; max;
min';
```

when making changes to one graph only, but

```
Crpel.GraphOptions[0] := 'Title; 0; Arial; H; T; F; legend; max;
min';
```

```
Crpel.GraphOptions[1] := 'Title; 1; Arial; H; T; F; legend; max;
min';
```

when making changes to more than one graph.

- *Section Codes (16-bit), Page 495*, or
- *Section Codes (32-bit), Page 494*.

Use the following chart as a guide in entering the required property values:

Parameter	Description	Values expected
sectionCode	Specifies the section in which you want to modify a graph.	Please refer to the section code table (see <i>Section Codes (32-bit), Page 494</i>).
graphNum	Specifies which graph in the section you want to modify.	Graphs in a section are numbered, starting with zero, left to right first, then top to bottom.
fontFace	Specifies the font face you want to use for the entire graph.	Actual name of font (i.e., Arial).
barDirection	In a bar graph, specifies the direction in which you want the graph bars to appear.	H = horizontal, V = vertical, X = as is
labelRisers	Specifies whether or not you want to show the data value on every riser.	T= true, F = False, X = as is
gridLines	Specifies whether or not you want to show grid lines.	T= true, F = False, X = as is
legend	Specifies whether or not you want to show a legend.	T= true, F = False, X = as is
max	Specifies the maximum value you want included in your graph.	Enter a number.
min	Specifies the minimum value you want included in your graph.	Enter a number.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

GraphText

Description

Sets the various text components for the specified graph.

Usage

```
[form.]Report.GraphText(ArrayIndex) [:=sectionCode; graphNum; title;  
subTitle; footnote; series; group; x; y; z];
```

For example:

```
Crpel.GraphText[0] := 'HEADER; 0;;;;;new x label; new y label;new z  
label';
```

«Resets the x, y, and z labels for the first graph in the Page Header section.»

Remarks

- Select your section code from the section code table for your operating system (see *Section Codes (32-bit)*, Page 494, or *Section Codes (16-bit)*, Page 495.).
- With GraphText, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action := 1” call (see *Action*, Page 740). The array index value for GraphText simply specifies the sequence number for the change. Thus:

```
Crpel.GraphText[0] := 'DETAIL; 1; title string;  
subtitle string;footnote string; series string;  
group string; x string; y string; z string';
```

when making changes to one graph only, but

```
Crpel.GraphText[0] := 'DETAIL 1; title string;  
subtitle string; footnote string; series string; group string; x  
string; y string; z string';  
  
Crpel.GraphText[1] := 'HEADER 1; title string;  
subtitle string; footnote string; series string; group string; x  
string; y string; z string';
```

when making changes to more than one graph.

- *title*, *subTitle*, *footnote*, *series*, *group*, *x*, *y*, and *z* are the strings you want to label the appropriate parts of the graph.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

GraphType

Description

Sets the kind of graph used in the selected section in the specified report.

Usage

```
[form.]Report.GraphType(ArrayIndex)
[:=sectionCode; graphNum; graphType];
```

For example:

```
Crpel.GraphType[0] := 'GH1; 0; PIE';
```

«Specifies a Pie graph as the first graph (graphNum := 0) in the Group Header 1 section.»

Remarks

- With GraphType, you can specify changes to one or more graphs at runtime. Those changes then take place sequentially when you make the “Action := 1“ call (see *Action, Page 740*). The array index value for GraphType specifies the sequence number for the change. Thus:

```
Crpel.GraphType[0] := 'DETAIL; 0; PIE';
```

when making changes to one graph only, but

```
Crpel.GraphType[0] := 'HEADER; 0; PIE';
```

```
Crpel.GraphType[1] := 'DETAIL; 0; PIE';
```

when making changes to more than one graph.

- Select sectionCode from the section code table (see *Section Codes (32-bit), Page 494*).
- Graph numbers are 0 origin; the first graph in a section is number 0, the second is number 1, etc.

- Multiple graphs in a section are numbered left to right first, then top to bottom.

Select from the following graph types for the GraphType value for this property:

<i>For this type of graph</i>	<i>Use this code for graphType</i>
Side-by-side	SIDEBYSIDE
3-D side-by-side	3DSIDE
Stacked bar	STACKEDBAR
3-D stacked bar	3DSTACKED
Percent bar	PERCENTBAR
3-D percent bar	3DPERCENT
Line	LINE
Area	AREA
3-D bars	3DBARS
Pie	PIE
Multiple pie	MULTIPLEPIE
Weighted pie	WEIGHTEDPIE

Data Type

Arrays of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

GroupCondition

Description

Specifies what kind of change in the Group Condition Field will trigger the creation of a group.

Usage

```
[form.]Report.GroupCondition(SequentialIndex)
[ := group;field;condition;sortDirection];
```

For example:

```
Crpe1.GroupCondition[0] := 'GROUP1 ; {header.ORDERNUM} ; ANYCHANGE ; A' ;
```

«Specifies that any change in the *ordernum* field in Group1 will trigger a new grouping.»

Remarks

Refer to the following tables for parameter values for this property:

Parameter	Description	Values expected
group	The group in which you want to set the group condition.	The outermost group on the report is GROUP1, the next group is GROUP2, etc.
field	The name of the field that triggers a grouping whenever its value changes.	Enter the name in the following format: {file.FIELDNAME}.condition. Enter the condition that triggers the grouping.
sortDirection	The direction in which groups are to be sorted.	A = Ascending, D = Descending

Condition (Date Fields)	Condition Code
Daily	DAILY
Weekly	WEEKLY
Bi-weekly	BIWEEKLY
Semi-monthly	SEMIMONTHLY
Monthly	MONTHLY
Quarterly	QUARTERLY
Semi-annually	SEMIANNUALLY
Annually	ANNUALLY

Condition (Boolean Fields)	Condition Code
To Yes	TOYES
To No	TONO
Every Yes	EVERYYES
Every No	EVERYNO
Next Is Yes	NEXTISYES
Next Is No	NEXTISNO

<i>Condition for all other data types</i>	
Any Change	ANYCHANGE

Data Type

Array of strings

Availability

Runtime only

Related Topics

Resetting Array Properties, Page 837

GroupSelectionFormula

Description

Specifies the groups to be used when printing the report.

Usage

```
[form.]Report.GroupSelectionFormula(Sequential Index)  
[:=GroupSelectionFormula];
```

Enter the group selection formula just as you would enter it in the Formula Editor.

For example:

```
Crpel.GroupSelectionFormula[0] := 'Count ({header.AMOUNT},  
{header.CUSTNAME}) > 20';
```

In this code the values in the Amount field are grouped each time the value in the CustName field changes and the values are counted. Then this property setting limits the report to only those groups for which the group count is greater than 20.

Remarks

If your group selection formula includes internal quotes, change all of the internal double quotes to single quotes and then surround the entire selection formula in double quotes.

NOTE: If you have created a group selection formula in your report at design time, any group selection formula you enter here will be appended to that group selection formula, connected by an "and". Thus, your records will be selected based on a combination of the two formulas.

Data Type

TCrpeString

Availability

Design time; Runtime

Related Topics

Resetting TCrpe String Properties, Page 838

GroupSortFields

Description

Specifies the group field(s) that are to be used to sort your data when the report is printed.

Usage

```
[form.]Report.GroupSortFields(ArrayIndex)
[ := "+|-}GroupField" ];
```

Enter the group field(s) on which you want your report to be sorted.

For example, assume that you have broken your data into state groups and had Seagate Crystal Reports count the number of customers in each group. In order to print the group with the highest count first, then the group with the next highest count, etc. (descending order), enter a string similar to the following:

```
Crpel.GroupSortFields[0] := '-Count
({customer.CUSTOMER_ID},{customer.REGION})';
```

Remarks

- Use a separate line of code to specify each group sort field.
- Enter group sort fields in the order that you want them to sort your report. For example, if you want your report to be sorted first on group sort field A and then on group sort field B, specify group sort field A in your first line of code and group sort field B in your second line of code.
- The first group sort field you specify must be assigned array index 0, the second group sort field must be assigned array index 1, etc.
- The index values you assign must be continuous; no gaps are allowed (0, 1, 2 = OK, 0, 1, 3 = wrong).

- Array index values must be subscripted in the code immediately after the property name (i.e., Crpe1.GroupSortFields[0]).
- If you have specified sort fields for your report at design time, any sort fields you enter here will replace the sort fields in your report.
- If you do not use this property, the program will use the sorting instructions that you specified in the report.
- If you want to clear the group sort fields in your report, use an empty string (i.e., Crpe1.GroupSortFields[0] := ""); Use this only if you are certain that GroupSortFields has already been set. If you're unsure use the Crpe1.Clear property or this code:

```
If Crpe1.GroupSortFields[0] <> '' then
  Crpe1.GroupSortFields[0] := '';
```

NOTE: The group sort field entry must follow the sort direction sign (+ or -) with no intervening space.

NOTE: To find the correct syntax for any group in your report using Seagate Crystal Reports for Delphi:

- select the **Insert|Formula Field command**,
- enter any formula name in the **Insert Formula dialog box** when it appears,
- click the scroll button on the **Field**.

*Seagate Crystal Reports enters the group field name in the **Formula Text box**. Use the name and syntax from that text box when constructing your group sort field string.*

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

LastErronumber

Description

Returns the error code for the last runtime error.

Usage

```
[form.]Report.LastErrorNumber;
```

For example:

```
ShowMessage (IntToStr(Crpe1.LastErrorNumber));
```

«If an error occurs, this code stores the error code as a number.»

Remarks

LastErrorMessage must be called after *Action*, *Page 740*, in order to display relevant values. After you have printed your report, you can refer to this property to get an error number (if any). If there was no error in printing, LastErrorMessage = 0.

Data Type

Integer

Availability

Runtime

LastErrorMessage

Description

Returns the error string for the last runtime error.

Usage

```
[form.]Report.LastErrorMessage;
```

For example:

```
ShowMessage (Crpe1.LastErrorMessage);
```

«If an error occurs, this code calls up a message box that displays the error string.»

Remarks

LastErrorMessage must come after *Action*, *Page 740*, in order to display relevant values. After you have printed your report, you can refer to this property to get an error string (if any). If there was no error in printing, LastErrorMessage, *Page 765*, is 0.

Data Type

String

Availability

Runtime

LogOnInfo

Description

Logs on to one or more SQL servers or a password-protected databases.

Usage

```
[form.]Report.LogOnInfo[ArrayIndex] := 'DSN = Datasource;UID =
UserID;PWD = Password;DSQ = Database';

crpel.LogOnInfo[0] := 'DSN = Accounting;UID = 734;PWD = bigboard;DSQ =
Administration';

«Connects to the “Administration” database on the “Accounting” server using the user ID #734
and the password “bigboard”.»

crpel.LogOnInfo[0] := 'dsn=;uid=;pwd=bigboard;dsq=';

«Connects to a password-protected Paradox database. All that is being passed is the password
bigboard.»
```

Remarks

- Use a separate line of code for each table for which you want to change the logon information.
- The order of tables in the array must conform to the order of tables in the report. (You can use Database | Set Location to determine the order of tables in the report.)
- The first table in the report is array index (0), the second file is (1), etc.
For example, to change the logon information of the first and third tables in a report to the NEW server, use the following syntax:

```
CrystalReport1.LogOnInfo(0) = "DSN = NEW;UID = 734;PWD =
bigboard;DSQ = Administration1"CrystalReport1.LogOnInfo(2) =
"DSN = NEW;UID = 734;PWD = bigboard;DSQ = Administration2"
```

- Enter the parameters necessary to log on to each SQL server table that you need to change information for in your report.

- Parameters should be in the following format:

```
DSN = name;UID = userID;PWD = password;DSQ = database qualifier
```

- *name* is the server name,
- *userID* is the name you have been assigned for logging on to the SQL server,
- *password* is the password you have been assigned for logging on to the SQL server, and
- *database* qualifier is the database name if your server uses the database concept.

NOTE: This parameter is required only when it is applicable to the ODBC/SQL driver you are using.

NOTE: Before you can use this property for an ODBC/SQL database, you must install the ODBC/SQL driver for whatever SQL database you are planning to use, and put the Database/BIN location in your path.

Data Type

Array of strings

Availability

Runtime

Related Report Engine Functions

PESetNthTableLogOnInfo, Page 320

MarginBottom

Description

Sets the bottom margin for the specified report.

Usage

```
[form.]Report.MarginBottom[ := MarginSetting];
```

For example:

```
Crpel.MarginBottom := 720;
```

«Sets a 1/2 inch bottom margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips.

Data Type

Word

Availability

Design time; Runtime

MarginLeft

Description

Sets the left margin for the specified report.

Usage

```
[form.]Report.MarginLeft[:=MarginSetting];
```

For example:

```
Crpe1.MarginLeft := 1440;
```

«Sets a 1 inch left margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips.

Data Type

Word

Availability

Design time; Runtime

MarginRight

Description

Sets the right margin for the specified report.

Usage

```
[form.]Report.MarginRight[ :=MarginSetting];
```

For example:

```
Crpel.MarginRight := 1440;
```

«Sets a 1 inch right margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips.

Data Type

Word

Availability

Design time; Runtime

MarginTop

Description

Sets the top margin for the specified report.

Usage

```
[form.]Report.MarginTop[ :=MarginSetting];
```

For example:

```
Crpel.MarginTop := 720;
```

«Sets a 1/2 inch top margin for the report (1 inch = 1440 twips).»

Remarks

MarginSetting is the margin you want, in twips.

Data Type

Word

Availability

Design time; Runtime

MDIChild

Description

This property is used to make the preview window a child of the specified form.

Usage

```
[form.]Report.MDIChild[ := {True | False}] ;
```

For example:

```
Crpe1.MDIChild := True;
```

«The preview window will be a child window of the current form.»

Remarks

Make sure that the form's FormStyle property is set to fsMDIForm when you set MDIChild to True.

Data Type

Boolean

Availability

Design time; Runtime

NumberOfGroups

Description

Returns the number of groups in the report.

Usage

```
[form.]Report.NumberOfGroups;
```

For example:

```
NumberOfGroups := Crpel.NumberOfGroups;
```

Data Type

Smallint

Availability

Runtime

ParamFields

Description

Changes the default value of the specified parameter field. When the prompting dialog box appears for the parameter field, the value you specify with this property is the value you are prompted with.

Usage

```
[form.]Report.ParamFields[ArrayIndex][ := "ParameterName";  
NewValue; SetCurrentValue"];
```

Remarks

- The parameter, SetCurrentValue can either be set to TRUE or FALSE.
- If set to TRUE, the parameter value is passed to the current value in the report; the user is not prompted to enter the parameter value.
- If set to FALSE, the parameter value is passed to the default value for the parameter; the user is prompted to enter the parameter value, with the value you set showing as the default value.

- The default value for SetCurrentValue is FALSE.
- This property does not eliminate the prompt by specifying a current value for the parameter field. You will still be prompted but the default value in the prompt will be the value you specify.
- Use a separate line of code for each parameter field for which you want to change the value.
- The order of values in the array must conform to the order of parameter fields in the report.
- The first parameter field in the report is array index (0), the second is (1), etc.
- For example, to change the value of the first and third parameter fields in a report (parameter1 and parameter3) to “red” and “blue” respectively use the following syntax:

```
Crpel.ParamFields[0] := "parameter1;red";
Crpel.ParamFields[2] := "parameter3;blue";
```

Data Type

Array of strings

Availability

Runtime only

Password

Description

Enters the password needed to use database tables on a restricted Access .mdb file.

Usage

```
[form.]Report.Password[:=Password];
```

For example:

```
Crpel.Password := 'dogsncats';
```

«Enters the password *dogsncats*.»

Remarks

Enter the password you have been assigned.

Data Type

String

Availability

Runtime

PrintControls

Description

Method to show/hide the standard button bar.

Usage

```
[form.]Report.PrintControls[ :={True|False}];
```

For example:

```
Crpe1.PrintControls:=True;
```

Remarks

This property can only be used after using *Action*, *Page 740*, or *Execute*, *Page 753*, to preview the report. To show or hide the controls before the preview window is actually displayed, use *WindowControls*, *Page 816*.

Data Type

Boolean

Availability

Runtime

PrintDay

Description

Sets the day component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintDay[ :=Day];
```

For example:

```
Crpel.PrintDay := 23;
```

«Sets 23 as the print day.»

Remarks

- Enter a value from 1 to 31.
- *PrintYear*, *Page 795*, *PrintMonth*, *Page 794*, and *PrintDay* function together. You must change the value of all three to change the print date. If you do not change all three, the print date saved with the report is used. This may be the current date if a specific date is not saved with the report.

Data Type

Word

Availability

Design time; Runtime

PrintEnded

Description

Indicates whether or not the printing process is completed.

Usage

```
[form.]Report.PrintEnded;
```

For example:

```
Crpel.Destination := toPrinter;  
Crpel.Execute  
While Not Crpel.PrintEnded Do  
  Application.ProcessMessages;  
Close;
```

«Sends the report to the printer. Will not allow the program to close the current form until the printing is finished.»

Remarks

- The value of this property is True when the Engine is done running the report.
- When destination is set to Window, this property will not return True until you have gone to the last page of the report.

Data Type

Boolean

Availability

Runtime

PrinterCollation

Description

If you specify more than one copy to be printed (using *PrinterCopies*, Page 777), PrinterCollation specifies whether or not the copies will be collated.

Usage

```
[form.]Report.PrinterCollation[ := ]TCollation;
```

For example:

```
Crpel.PrinterCollation := Collated;
```

«Collates the copies of the specified report.»

Data Type

TCollation = (Uncollated, Collated, DefaultCollation);

Availability

Design time; Runtime

PrinterCopies

Description

Sets the number of report copies to be printed.

Usage

```
[form.]Report.PrinterCopies[:=NumCopies];
```

For example:

```
Crpel.PrinterCopies := 3;
```

«Specifies that the program is to print three copies of the report.»

Remarks

The number used for PrinterCopies must not be zero or a negative value.

Data Type

Word

Availability

Design time; Runtime

PrinterDriver

Description

Sets the name of the printer driver that is to print the report.

Usage

```
[form.]Report.PrinterDriver [:=DriverName];
```

For example:

```
Procedure TForm1.Button1.Click(Sender: TObject);
begin
  var
    lpPrinter, lpDriver, lpPort : pChar;
```

```

    hMode : Thandle;
begin
  lpPrinter := StrAlloc(255);
  try
    lpDriver := StrAlloc(255);
    try
      lpPort := StrAlloc(255);
      try
        Cancel := not
        PrintDialog1.Execute;
        Printer.GetPrinter(lpPrinter,
                           lpDriver, lpPort, hMode);
        Crpel.PrinterName :=
          StrPas(lpPrinter);
        Crpel.PrinterDriver :=
          StrPas(lpDriver);
        Crpel.PrinterPort := StrPas(lpPort);
        Crpel.PrinterMode := hMode;
      finally
        StrDispose(lpPort);
      end;
    finally
      StrDispose(lpDriver);
    end;
  finally
    StrDispose(lpPrinter);
  end;
end;

```

Remarks

PrinterDriver, *PrinterMode*, *Page 779*, *PrinterName*, *Page 780*, and *PrinterPort*, *Page 781*, all work together to define the printer that the report is to be sent to. All four properties must be set in order to define a new printer. If all four properties are not set, the printer defined in the report will be used. This may be the users default printer if none has been specified in the report.

Data Type

String

Availability

Design time; Runtime

PrinterMode

Description

Allows you to set the MS Device Mode for a report (orientation, paper size, etc.).

Usage

```
[form.]Report.PrinterMode[ := Thandle];
```

For example:

```
Procedure TForm1.Button1.Click(Sender: TObject);
var
    lpPrinter, lpDriver, lpPort : pChar;
    hMode : Thandle;
begin
    lpPrinter := StrAlloc(255);
    try
        lpDriver := StrAlloc(255);
        try
            lpPort := StrAlloc(255);
            try
                Cancel := not
                PrintDialog1.Execute;
                Printer.GetPrinter(lpPrinter,
                    lpDriver, lpPort,
                    hMode);
                Crpel.PrinterName :=
                StrPas(lpPrinter);
                Crpel.PrinterDriver :=
                StrPas(lpDriver);
                Crpel.PrinterPort := StrPas(lpPort);
                Crpel.PrinterMode := hMode;
                finally
                    StrDispose(lpPort);
            end;
            finally
                StrDispose(lpDriver);
            end;
            finally
                StrDispose(lpPrinter);
            end;
    end;
```

Remarks

PrinterDriver, *Page 777*, *PrinterMode*, *PrinterName*, *Page 780*, and *PrinterPort*, *Page 781*, all work together to define the printer that the report is to be sent to. All four properties must be set in order to define a new printer. If all four properties are not set, the printer defined in the report will be used. This may be the users default printer if none has been specified in the report.

Data Type

THandle

Availability

Design time; Runtime

PrinterName

Description

Sets the name of the printer that is to print the report.

Usage

```
[form.]Report.PrinterName[ :=PrinterName];
```

For example:

```
Procedure TForm1.Button1.Click(Sender: TObject);
var
  lpPrinter, lpDriver, lpPort : pChar;
  hMode : THandle;
begin
  lpPrinter := StrAlloc(255);
  try
    lpDriver := StrAlloc(255);
    try
      lpPort := StrAlloc(255);
      try
        Cancel := not
        PrintDialog1.Execute;
        Printer.GetPrinter(lpPrinter,
                           lpDriver, lpPort,
                           hMode);
      Crpel.PrinterName :=
```

```

        StrPas(lpPrinter);
        Crpel.PrinterDriver := 
        StrPas(lpDriver);
        Crpel.PrinterPort := StrPas(lpPort);
        Crpel.PrinterMode := hMode;
        finally
        StrDispose(lpPort);
    end;
    finally
        StrDispose(lpDriver);
    end;
    finally
        StrDispose(lpPrinter);
    end;
end;

```

Remarks

PrinterDriver, *Page 777*, *PrinterMode*, *Page 779*, *PrinterName* and *PrinterPort*, *Page 781*, all work together to define the printer that the report is to be sent to. All four properties must be set in order to define a new printer. If all four properties are not set, the printer defined in the report will be used. This may be the users default printer if none has been specified in the report.

Data Type

String

Availability

Design time; Runtime

PrinterPort

Description

Sets the port for the specified printer.

Usage

```
[form.]Report.PrinterPort[ :=PortName];
```

For example:

```
Procedure TForm1.Button1.Click(Sender: TObject);  
  
var  
    lpPrinter, lpDriver, lpPort : pChar;  
    hMode : Thandle;  
begin  
    lpPrinter := StrAlloc(255);  
    try  
        lpDriver := StrAlloc(255);  
        try  
            lpPort := StrAlloc(255);  
            try  
                Cancel := not  
                PrintDialog1.Execute;  
                Printer.GetPrinter(lpPrinter,  
                    lpDriver, lpPort,  
                    hMode);  
                Crpel.PrinterName :=  
                    StrPas(lpPrinter);  
                Crpel.PrinterDriver :=  
                    StrPas(lpDriver);  
                Crpel.PrinterPort := StrPas(lpPort);  
                Crpel.PrinterMode := hMode;  
                finally  
                    StrDispose(lpPort);  
                end;  
                finally  
                    StrDispose(lpDriver);  
                end;  
                finally  
                    StrDispose(lpPrinter);  
                end;  
            finally  
                StrDispose(lpPort);  
            end;  
        finally  
            StrDispose(lpDriver);  
        end;  
    finally  
        StrDispose(lpPrinter);  
    end;  
end;
```

Remarks

PrinterDriver, *Page 777*, *PrinterMode*, *Page 779*, *PrinterName*, *Page 780*, and *PrinterPort* all work together to define the printer that the report is to be sent to. All four properties must be set in order to define a new printer. If all four properties are not set, the printer defined in the report will be used. This may be the users default printer if none has been specified in the report.

Data Type

String

Availability

Design time; Runtime

PrinterStartPage

Description

Sets the first page to be printed.

Usage

```
[form.]Report.PrinterStartPage[ :=StartPage];
```

For example:

```
Crpel.PrinterStartPage := 7;
```

«Specifies that printing is to begin with Page 7 of the report.»

Remarks

If a value less than or equal to 0 is used for PrinterStartPage, the value is ignored and printing starts with Page 1.

Data Type

Word

Availability

Design time; Runtime

PrinterStopPage

Description

Sets the last page to be printed.

Usage

```
[form.]Report.PrinterStopPage[ :=StopPage];
```

For example:

```
Crpe1.PrinterStopPage := 12;
```

«Specifies that the printing is to end with Page 12 of the report.»

Remarks

Use a value of -1 for PrinterStopPage to indicate that printing is to continue through to the last page.

Data Type

Word

Availability

Design time; Runtime

PrintFileCharSepQuote

Description

Sets the quote character used to enclose alphanumeric field data when printing to a file using character-separated export format.

Usage

```
[form.]Report.PrintFileCharSepQuote[ :=Quote];
```

For example:

```
Crpe1.PrintFileCharSepQuote := `";
```

«Uses the quotation character (") to surround values saved in character-separated export format.»

Remarks

- Applies only when *PrintFileType*, *Page 791*, is CharacterSeparated values.
- Applies only when *Destination*, *Page 744*, is toFile, toEMailviaMAPI, or toEMailviaVIM.
- If you assign a string to PrintFileCharSepQuote that is longer than one character, the VCL uses only the first character of that string. For example, if you assign 'quote' to the property, the VCL will only recognize 'q'.

Data Type

String

Availability

Design time; Runtime

PrintFileCharSepSeparator

Description

Sets the character(s) you want to use to separate the fields when printing to a file using the Character Separated Value format.

Usage

```
[form.]Report.PrintFileCharSepSeparator[ := Separator];
```

For example:

```
Crpe1.PrintFileCharSepSeparator := '@';
```

«Specifies that the “@” character is to be used for separating field values.»

Remarks

- Applies only when *PrintFileType*, [Page 791](#), is CharacterSeparated values.
- Applies only when *Destination*, [Page 744](#), is set to toFile, toEMailViaMAPI, or toEMailViaVIM.

Data Type

String

Availability

Design time; Runtime

PrintFileLinesPerPage

Description

Indicates the number of lines to be printed before the page break. The default is 60 lines.

Usage

```
[form:] Report.PrintFileLinesPerPage [:=NumLines]  
crpel.PrintFileLinesPerPge := 50  
«Fifty lines will be printed before a page break.»
```

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEExportTo, Page 146, with PEExportOptions.FormatOptions set to an UXFODBCOptions structure

PrintFileName

Description

Specifies the name of the file to which the report is to be printed.

Usage

```
[form.]Report.PrintFileName[:=FileName];
```

For example:

```
Crpel.PrintFileName := 'c:\crw\cust_rpt.txt';  
«Prints the report to a file named "cust_rpt.txt" in the C:\CRW directory.»
```

Remarks

- You can double-click this property or click the ellipsis (...) in the Settings box to call up the Choose Print Filename dialog box. In that dialog box, select the name of the file and the path to which you want the program to print the report.
- Select a value for this property only if you are printing to a file, if the value you assigned to the *Destination*, Page 744, is toFile.

NOTE: If you want to specify the *PrintFileName* at runtime, make certain that you enclose it in quotes in your code.

- **Records**
Record style (columns of values). Does not use commas or separators. Outputs every record with a fixed field width.
- **TabSeparated**
Tab separated values. Presents data in tabular form. Encloses alphanumeric field data in quotes and separates fields with tabs.
- **ASCII**
Text style. Saves the data in ASCII text format with all values separated by spaces. This style looks most like the printed page.
- **DIF**
Saves the data in DIF (data interchange format) format. This format is often used for the transfer of data between different spreadsheet programs.
- **CSV**
Comma separated values. Encloses alphanumeric field data in quotes and separates fields with commas.
- **CharacterSeparated**
Saves the data as character separated values in ASCII text format. All values are separated by a character or characters specified by *PrintFileCharSepSeparator*, *Page 785*.
- **TabSeparatedText**
Saves the data in ASCII text format with all values separated by tabs.
- **CrystalReportsRPT**
Standard Seagate Crystal Reports format is used. Most often used for sending the report to another user via e-mail.
- **Excel2**
Exports the report as a Microsoft Excel 2.1 Worksheet.
- **Excel4**
Exports the report as a Microsoft Excel 4.0 Worksheet.
- **LotusWK1**
Exports the report as a Lotus 1-2-3 WK1 format worksheet.
- **LotusWK3**
Exports the report as a Lotus 1-2-3 WK3 format worksheet.
- **LotusWKS**
Exports the report as a Lotus 1-2-3 WKS format worksheet.
- **RTF**
Saves the data in Rich Text Format.
- **WordForDOS**
Uses the Microsoft Word for DOS format to save the data in the report.
- **WordForWindows**
Uses the Microsoft Word for Windows format to save the data in the report.

— **WordPerfect**

Uses WordPerfect format to save the data in the report.

Data Type

String

Availability

Design time; Runtime

PrintFileODBCPassword

Description

Used whenever you want to export in ODBC format. Specifies the Password that you need to connect to the data source.

Usage

```
[form.]Report.PrintFileODBCPassword[ := 'Password' ];
```

For example:

```
crpe1.PrintFileODBCPassword := 'merry%%5';
```

«merry%%5 is the name of the password to connect to the data source.»

Remarks

This is only required if the ODBC data source that you are exporting to requires a password.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.FormatOptions set to an UXFODBCOptions structure.

PrintFileODBCSource

Description

Used whenever you export in ODBC format. Specifies the name of the data source that you want to export to.

Usage

```
[form.]Report.PrintFileODBCSource[ := 'DataSource'];
```

For example:

```
crp1.PrintFileODBCSource := 'pickle';
```

«pickle is the name of the data source that you want to export to.»

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.FormatOptions set to an UXFODBCOptions structure.

PrintFileODBCTable

Description

Used whenever you want to export in ODBC format. Specifies the name of the table you want to export to in the data source.

Usage

```
[form.]Report.PrintFileODBCTable[ := 'TableName'];
```

For example:

```
crpe1.PrintFileODBCTable := 'Employees';  
«Employees is the name of the table in the data source to export to.»
```

Data Type

String

Availability

Design Time; Runtime

PrintFileODBCUser

Description

Used whenever you export in ODBC format. Specifies the User ID that you need to connect to the data source.

Usage

```
[form.]Report.PrintFileODBCUser[ := 'UserID'];
```

For example:

```
crpe1.PrintFileODBCUser := 'LisaB';  
«LisaB is the User ID needed to connect to the data source.»
```

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEExportTo, *Page 146*, with PEExportOptions.FormatOptions set to an UXFODBCOptions structure.

PrintFileType

Description

Specifies the type of print file used when printing a report to a file.

Usage

```
[form.]Report.PrintFileType[ :=FileType];
```

For example:

```
Crpel.PrintFileType := TabSeparated;  
«Prints the report to a file in a tab separated format.»
```

Remarks

Select one of the following print file types if you are printing to a file, if the value you assigned to the *Destination*, *Page 744*, is toFile.

- **Records**

Record style (columns of values). Does not use commas or separators. Outputs every record with a fixed field width.

- **TabSeparated**

Tab separated values. Presents data in tabular form. Encloses alphanumeric field data in quotes and separates fields with tabs.

- **ASCII**

Text style. Saves the data in ASCII text format with all values separated by spaces. This style looks most like the printed page.

- **DIF**

Saves the data in data interchange format. This format is often used for the transfer of data between different spreadsheet programs.

- **CSV**

Comma separated values. Encloses alphanumeric field data in quotes and separates fields with commas.

- **CharacterSeparated**

Saves the data as character separated values in ASCII text format. All values are separated by a character or characters specified by *PrintFileCharSepSeparator*, *Page 785*.

- **TabSeparatedText**

Saves the data in ASCII text format with all values separated by tabs.

- **CrystalReportsRPT**
Standard Seagate Crystal Reports format is used. Most often used for sending the report to another user via e-mail.
- **Excel2**
Exports the report as a Microsoft Excel 2.1 Worksheet.
- **Excel3**
Exports the report as a Microsoft Excel 3.0 Worksheet.
- **Excel4**
Exports the report as a Microsoft Excel 4.0 Worksheet.
- **LotusWK1**
Exports the report as a Lotus 1-2-3 WK1 format worksheet.
- **LotusWK3**
Exports the report as a Lotus 1-2-3 WK3 format worksheet.
- **LotusWKS**
Exports the report as a Lotus 1-2-3 WKS format worksheet.
- **QuattroPro5**
Exports the report as a Quattro Pro 5.0 WB1 format file.
- **RTF**
Saves the data in Rich Text Format.
- **WordForDOS**
Uses the Microsoft Word for DOS format to save the data in the report.
- **WordForWindows**
Uses the Microsoft Word for Windows format to save the data in the report.
- **WordPerfect**
Uses WordPerfect format to save the data in the report.

Data Type

Integer (Enumerated)

Availability

Design time; Runtime

PrintFileUseRptDateFmt

Description

When printing to a file, indicates whether or not the program should save dates in the same date format (MDY, DMY, etc.) that is used in the report or instead, optimize the dates for the file format you have selected.

Usage

```
[form.]Report.PrintFileUseRptDateFmt [ := {True|False} ];
```

For example:

```
Crp1.PrintFileUseRptDateFmt := True;
```

«Specifies that the program should print dates in the same format as used in the report.»

Remarks

Applies only when *Destination*, *Page 744*, is set to toFile, toEMailViaMAPI, or toEMailViaVIM.

Data Type

Boolean

Availability

Design time; Runtime

PrintFileUseRptNumberFmt

Description

When printing to a file, indicates whether or not the program should print numbers in the same format (decimal places, negatives, etc.) that you have used in the report or instead, optimize the numbers for the file format you have selected.

Usage

```
[form.]Report.PrintFileUseRptNumberFmt [ := {True|False} ];
```

For example:

```
Crpe1.PrintFileUseRptNumberFmt := True;
```

«Specifies that the program should print numbers in the same format as used in the report.»

Remarks

- Applies only when *PrintFileType*, *Page 791*, is set to Records, TabSeparated, DIF, CSV, or CharacterSeparated.
- Applies only when *Destination*, *Page 744*, is set to toFile, toEMailViaMAPI, or toEMailViaVIM.

Data Type

Boolean

Availability

Design time; Runtime

PrintMonth

Description

Sets the month component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintMonth[ :=Month];
```

For example:

```
Crpe1.PrintMonth := 7;
```

«Sets July as the print month.»

Remarks

- Enter a value from 1-12 where January = 1, December = 12.
- *PrintYear*, *Page 795*, *PrintMonth*, and *PrintDay*, *Page 774*, all function together. You must change the value of all three to change the print date. If you do not change all three, the print date saved with the report is used. This may be the current date if a specific date is not saved with the report.

Data Type

Word

Availability

Design time; Runtime

PrintYear

Description

Sets the year component of the print date (if different from the actual date the report is printed).

Usage

```
[form.]Report.PrintYear[ :=Year];
```

For example:

```
Crpe1.PrintYear := 1994;
```

«Sets the year component of the print date to 1994.»

Remarks

- Enter the print year as a four-digit number.
- PrintYear, *PrintMonth*, *Page 794*, and *PrintDay*, *Page 774*, all function together. You must change the value of all three to change the print date. If you do not change all three, the print date saved with the report is used. This may be the current date if a specific date is not saved with the report.

Data Type

Word

Availability

Design time; Runtime

ProgressDialog

Description

Enables/disables the display of the Progress dialog box. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

Usage

```
[form.]ReportProgressDialog[ := {True|False}];
```

For example:

```
Crp1ProgressDialog := False;
```

«Turns off the Progress dialog box that usually appears during exporting or printing.»

Remarks

Use this property to indicate whether or not a Progress dialog box should be displayed while the report is printed or exported. This property is set to True by default.

Data Type

Boolean

Availability

Design Time; Runtime

RecordsPrinted

Description

Determines the number of records actually printed.

Usage

```
[form.]Report.RecordsPrinted;
```

For example:

```
Printed := Crpel.RecordsPrinted;
```

«Fetches the number of records printed and stores it in the *Printed* variable.»

Remarks

If the report being printed contains one or more group selection formulas, the value returned by *RecordsPrinted* may be significantly less than the value returned by *RecordsSelected*, *Page 798*. Otherwise, this value should equal *RecordsSelected*.

Data Type

LongInt

Availability

Runtime

RecordsRead

Description

Specifies the number of records actually processed by the print job process.

Usage

```
[form.]Report.RecordsRead;
```

For example:

```
ReadSoFar := Crpel.RecordsRead;
```

«Fetches the number of records read so far and stores it in the *ReadSoFar* variable.»

Data Type

Long

Availability

Runtime

RecordsSelected

Description

Determines the number of records selected for inclusion in the report out of the total number of records read.

Usage

```
[form.]Report.RecordsSelected;
```

For example:

```
Selected := Crpel.RecordsSelected;
```

«Fetches the number of records selected and saves it in the *Selected* variable.»

Remarks

RecordsSelected will return a value anywhere between zero and the value returned by the RecordsRead property. The value returned by RecordsSelected depends on the queries and selection formulas set up in the report.

Data Type

LongInt

Availability

Runtime

ReportDisplayPage

Description

Indicates which page of a multi-page report is currently being displayed in the preview window.

Usage

```
[form.]Report.ReportDisplayPage;
```

For example:

```
Result := Crpel.DisplayPage;
```

«Fetches the number of the displayed page and stores it in the *Result* variable.»

Data Type

Word

Availability

Runtime

ReportLatestPage

Description

Determines the last page printed in the specified report.

Usage

```
[form.]Report.ReportLatestPage;
```

For example:

```
Latest := Crpel.ReportLatestPage;
```

«Fetches the number of the last page printed and stores it in the *Latest* variable.»

Data Type

Word

Availability

Runtime

ReportName

Description

Specifies the report to be printed.

Usage

```
[form.]Report.ReportName[ :=ReportName];
```

For example:

```
Crpel.ReportName := 'c:\crw\company.rpt';
```

«Prints the report named “company.rpt” that is located in the C:\CRW directory.»

Remarks

You can double-click this property or click the ellipsis (...) in the Object Inspector to call up the Choose Report File dialog box. In that dialog box, select the name and path of the report you want the program to print in response to an event.

NOTE: If you want to specify the ReportName at runtime, make certain that you enclose it in quotes in your code.

Data Type

String

Availability

Design time; Runtime

ReportPrinterSetup

Description

This property is used to set to enable or disable various settings made in the Seagate Crystal Reports Printer Setup dialog box (File menu).

Usage

```
[form.]Report.UserPrinterSetup.Enabled  
[:= {True|False}];  
[form.]Report.UserPrinterSetup.ReportPrinterSetup[ :=[TReportPrinterSetup]];
```

For example:

```
Crpel.UserPrinterSetup.Enabled := True;  
Crpel.UserPrinterSetup.ReportPrinterSetup := [rsOrientation];
```

«Preserves the orientation saved in the report.»

Remarks

UserPrinterSetup property must be enabled before you can use ReportPrinterSetup.

Data Type

TReportPrinterSetup; TReportPrinterSetups = set of [rsOrientation, rsPaperSize, rsPaperSource];

Availability

Design time; Runtime

ReportStartPage

Description

Determines the first page printed in the specified report.

Usage

```
[form.]Report.ReportStartPage;
```

For example:

```
StartPage := Crp1.ReportStartPage;
```

«Fetches the number of the first page printed and stores it in the *StartPage* variable.»

Data Type

Word

Availability

Runtime

ReportTitle

Description

Sets a report title that supplements the DOS report name at design time or runtime.

Usage

```
[form.]Report.ReportTitle[ :=Title];
```

For example:

```
Crpel.ReportTitle := 'Monthly Sales Report';
```

«Sets the report title to be Monthly Sales Report.»

Remarks

The value of the ReportTitle is not displayed in any part of the report, but is provided for your own use.

Data Type

String

Availability

Design time; Runtime

RetrieveTableData

Description

A read-only property array used by the RetrieveDataFiles method to store the table locations for the current report.

Usage

```
[form.]Report.RetrieveTableData[Index];
```

For example:

```
listbox1.items.add (crpel.RetrieveTableData[Index]);
```

«Adds the table at the specified index to the list box.»

Data Type

String

Availability

Runtime

SectionFont

Description

Sets the font for one or more sections in the specified report.

Usage

```
[form.]Report.SectionFont(SequentialIndex) [:=sectionCode; fontName;  
size; italic; bold; underline; strikethru];
```

For example:

```
Crpel.SectionFont[0] := 'Footer;Arial;12;N;N;N;Y';
```

«Sets the font for the footer section to 12 point, Arial, strikethrough.»

Remarks

With SectionFont, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make *Action, Page 740*, equal to 1. The array index value for SectionFont simply specifies the sequence number for the change. Thus:

```
Crpel.SectionFont[0] := 'DETAIL;Arial;12;N;N;N;Y';
```

when making changes to the DETAIL section only, but

```
Crpel.SectionFont[0] := 'HEADER;Arial;12;N;N;N;Y';
```

```
Crpel.SectionFont[1] := 'DETAIL;Arial;12;N;N;N;Y';
```

when making changes to more than one section.

Use the following table as a guide in supplying the required values for this property:

Parameter	Data type	Value expected
sectionCode	string	Please refer to the section code table (see <i>Section Codes (16-bit), Page 495</i> , or <i>Section Codes (32-bit), Page 494</i>).
fontName	string	The actual font name, i.e., Arial or Helvetica.

Parameter	Data type	Value expected
size	number	The size of the font in points (i.e., 12 or 16).
italic	character	T = true, F = False, X = as is [*]
bold	character	T = true, F = False, X = as is [*]
underline	character	T = true, F = False, X = as is [*]
strikethru	character	T = true, F = False, X = as is [*]

*X (as is) uses the value saved with the report.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

SectionFormat

Description

Sets the format for one or more sections in the specified report.

Usage

```
[form.]Report.SectionFormat(SectionArrayIndex)[ :=sectionCode;
visible; newPageBefore; newPageAfter; keepTogether; suppressBlank;
resetPageNAfter; printAtPageBottom; backgroundColor; underlay];
```

For example:

```
Crp1.SectionFormat[0]:= 'GH2;F;X;X;X;X;255.0.0;X';
```

«Hides the Group Header 2 section (visible = F) and changes the background color to red while maintaining default settings for all other switches.»

Remarks

With SectionFormat, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make *Action*, *Page 740*, equal to 1. The sequential index value for SectionFormat simply specifies the sequence number for the change. Thus:

```
Crpel.SectionFormat[0] := 'DETAIL;T;F;F;X;X;X;255.0.0;X';
```

when making changes to the DETAIL section only, but

```
Crpel.SectionFormat[0] := 'HEADER;T;F;F;X;X;X;255.0.0;X';
```

```
Crpel.SectionFormat[1] := 'DETAIL;T;F;F;X;X;X;255.0.0;X';
```

when making changes to more than one section.

Use the following table as a reference when entering parameter values for this property:

Parameter	Expected value
sectionCode	Please refer to the section code table (see <i>Section Codes (16-bit)</i> , <i>Page 495</i> , or <i>Section Codes (32-bit)</i> , <i>Page 494</i>).
visible	T = true, F = False, X = as is*
newPageBefore	T = true, F = False, X = as is*
newPageAfter	T = true, F = False, X = as is*
backgroundColor	Supply a RGB (Red, Green, Blue) value in the following format: <R>. <G>. where R, G, and B are each integers with a range from 0 to 255. For example: 189.210.100 If you do not want to change the color, do not place anything in this parameter.
underlaySection	T = true, F = False, X = as is*
keepTogether	T = true, F = False, X = as is*
suppressBlank	T = true, F = False, X = as is*
resetPageNAfter	T = true, F = False, X = as is*
printAtPageBottom	T = true, F = False, X = as is*

*X (as is) uses the settings saved with the report.

Data Type

Arrays of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

SectionLineHeight

Description

Specifies the line height in twips. A twip is 1/1440 inch; there are 20 twips in a point.

Usage

```
[form.]Report.SectionLineHeight(SequentialIndex)[ :=sectionCode; line;
height; ascent];
```

For example:

```
Crpel.SectionLineHeight[0] := 'GH0; 1; 500; 300';
```

«Sets the line height for the second line in the group header zero section to a height of 500 twips with an ascent of 300 twips.»

Remarks

With SectionLineHeight, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make *Action, Page 740*, equal to 1. The sequential index value for SectionLineHeight simply specifies the sequence number for the change. Thus:

```
Crpel.SectionLineHeight[0] := 'DETAIL;1;500;300';
```

when making changes to the DETAIL section only, but

```
Crpel.SectionLineHeight[0] := 'HEADER;1;500;300';
Crpel.SectionLineHeight[1] := 'DETAIL;1;500;300';
```

when making changes to more than one section.

Use the following table as a guide in supplying the required values for this property:

Parameter	Explanation
sectionCode	Specifies the section code for the report section(s) for which you want to set a new line height. Please refer to the section code table (see <i>Section Codes (32-bit)</i> , Page 494).
lineN	Specifies the line(s) for which you want to set the line height. Line numbers in a section are 0 origin: the first line number is 0, the second is 1, etc.
height	Specifies the line height in twips. A twip is 1/1440 inch; there are 20 twips in a point.
ascent	Specifies the ascent in twips. Ascent is the distance from the top of the allotted line space (line height) to the baseline of the font. The ascent parameter is used to specify the position of the baseline if you specify an oversized or undersized line height. If you set ascent to 0, the program puts the baseline at the top of the space; if you set ascent to the same value as height, the program sets the baseline at the bottom of the space. For any other baseline, specify the ascent in twips.

Data Type

Arrays of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

SectionMinHeight

Description

Sets the minimum section height for the specified report section.

Usage

```
[form.]Report.SectionMinHeight(SequentialIndex) [:=sectionCode;  
minHeight];
```

For example:

```
Crpe1.SectionMinHeight[0]:= 'ALL; 500';
```

«Sets the minimum height for all sections to 500 twips.»

Remarks

With `SectionMinHeight`, you can specify changes to one or more sections at runtime. Those changes then take place sequentially when you make *Action, Page 740*, equal to 1. The array index value for `SectionMinHeight` simply specifies the sequence number for the change. Thus:

```
Crpe1.SectionMinHeight[0] := 'DETAIL;500';
```

when making changes to the DETAIL section only, but

```
Crpe1.SectionMinHeight[0] := 'HEADER;500';
Crpe1.SectionMinHeight[1] := 'DETAIL;500';
```

when making changes to more than one section.

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

SelectionFormula

Description

Specifies the records to be used when printing the report. Whatever is set in this property gets appended to the original Selection Formula in the report. It will AND whatever is set to the original Selection Formula in the report.

Usage

```
[form.]Report.SelectionFormula(SequentialIndex) [:=SelectionFormula];
```

Enter the selection formula just as you would enter it in the Formula Editor in Seagate Crystal Reports. Search for the Formula Editor in Developer's online Help.

For example:

```
Crpe1.SelectionFormula[0] := '{file.COUNTRY} = "USA"';
```

This code limits the report to records that have the value 'USA' in the COUNTRY field (only United States records).

Remarks

- Make certain that you enclose your selection formula in single quotes.
- If your selection formula includes internal quotes, for example:

```
{file.STATE} = 'CA'
```

change all of the internal single quotes to double quotes and then surround the entire selection formula in single quotes as follows:

```
'{file.STATE} = "CA"'
```

- If you have created a selection formula in your report at design time, any selection formula you enter here will be appended to that selection formula. Thus, your records will be selected based on a combination of the two selection formulas.

Data Type

String

TCrpeString

Availability

Design time; Runtime

Related Topics

Resetting TCrpe String Properties, Page 838

SortFields

Description

Specifies the field(s) that are to be used to sort your data when the report is printed.

Usage

```
[form.]Report.SortFields(ArrayIndex)
[ := "+|-}SortField"];
```

Enter the fields on which you want the data in your report to be sorted.

For example, to sort an order database alphabetically by customer, and then by order date, you can enter code similar to the following:

```
Crpe1.SortFields[0] := '+{orders.CUSTOMER}';
Crpe1.SortFields[1] := '+{orders.ORDERDATE}';
```

Remarks

- Use a separate line of code to specify each sort field.
- Enter sort fields in the order that you want them to sort your report. For example, if you want your report to be sorted first on field A and then on field B, specify sort field A in your first line of code and sort field B in your second line of code.
- The sort field you specify must be assigned array index 0, the second sort field must be assigned array index 1, etc.
- The index values you assign must be continuous; no gaps are allowed (0, 1, 2 = OK, 0, 1, 3 = wrong).
- Array index values must be subscripted in the code immediately after the property name (i.e., Crpe1.SortFields[0]:=).
- If you have specified sort fields for your report at design time, any sort fields you enter here will replace the sort fields in your report.
- If you do not use this property, the program will use the sorting instructions that you specified in the report.
- If you want to clear the sort fields in your report, use an empty string (i.e., Crpe1.SortFields[0] := ''). Use this only if you are certain that SortFields has already been set. If you're unsure use the Crpe1.Clear property or this code:

```
If Crpe1.SortFields[0] <> '' then
    Crpe1.SortFields[0] := ''
```

- Enclose field names in braces.
- Sort fields can be database fields or formula fields. If you sort on a formula field, use the @ sign before the formula name (i.e., {@FORMULANAME}).

Data Type

Array of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

SQLQuery

Description

Sets the SQL query string used by the specified report.

Usage

```
[form.]Report.SQLQuery[SequentialIndex]  
[:=SQLQuery];
```

For example:

```
Crp1.SQLQuery[0] := 'SELECT authors.au_id, authors.au_lname,';  
Crp1.SQLQuery[1] := ' authors.au_fname FROM pubs2.dbo.authors';  
Crp1.SQLQuery[2] := ' authors WHERE authors.au_lname > ''Madison''';  
«Queries the SQL database to return only the records where the authors last name falls after  
Madison alphabetically.»
```

Remarks

- You may only change the WHERE and FROM sections of an SQL query. Although the property requires that you enter the entire SQL query, the SELECT section must not be different from the original query.
- If you are going to include an ORDER BY clause, you must precede it with CHR(13) and CHR(10).

Data Type

String

Availability

Design time; Runtime

Related Topics

Resetting TCrpe String Properties, Page 838

Status

Description

Determines the print status for the specified report.

Usage

```
[form.]Report.Status;
```

For example:

```
Status := Crpel.Status;
```

«Fetches the print status and saves it to the *Status* variable.»

Remarks

The Status property will return one of the following values:

- **0** - The report has not been printed or has not finished printing.
- **3** - The report has finished printing.
- **5** - The printing has been cancelled by the user.

When previewing a report, a status of 3 will only be returned when the user has previewed to the last page of the report.

Data Type

Integer

Availability

Runtime

StoredProcParam

Description

Sets the stored procedure parameters when using a report based on SQL stored procedures.

Usage

```
[form.]Report.StoredProcParam(Parameter Array Index) [:=newParameter];
```

For example:

```
Crpel.StoredProcParam[0] := '06/14/1989';
```

«Sets the first stored procedure parameter to the date June 14, 1989.»

Remarks

StoredProcParam sets the value of the specified parameter in an SQL database table that is based on a stored procedure. Pass the value you wish to set the parameter to as a string. If the parameter expects a different data type, you still must pass the value as a string. For example, to pass the integer value 396, use the string "396". The Crystal Report Engine will handle converting the value into integer format.

Data Type

Arrays of strings

Availability

Runtime

Related Topics

Resetting Array Properties, Page 837

Access Notes Sparse Array Properties, Page 837

UserName

Description

Enters the name given to a user for logging on to a protected Access .mdb file to obtain data files needed by the report.

Usage

```
[form.]Report.UserName[ :=Name];
```

For example:

```
Crpe1.UserName := 'MIS';
```

«Enters the user name "MIS".»

Remarks

- Enter the name you have been assigned.
- The name must be enclosed in quotes if the variable is being assigned at runtime.

Data Type

String

Availability

Runtime

WindowBorderStyle

Description

Specifies the type of border for the preview window.

Usage

```
[form.]Report.WindowBorderStyle [ :=TFormBorderStyle];
```

For example:

```
Crpe1.WindowBorderStyle := bsSizeable;
```

«Sets a sizeable border style for the preview window.»

Remarks

Select one of the following border styles for the preview window:

- bsNone (creates a window with no border).
- bsSingle (creates a window of a fixed size with a single line border).

- **bsSizeable** (creates a window that can be resized by the user).
- **bsDialog** (creates a window of fixed size with a double line border).

Select a value here only if you are printing to a window, if *Destination*, *Page 744*, is set to *toWindow*.

Data Type

Integer (Enumerated)

Availability

Design time; Runtime

WindowControlBox

Description

Specifies whether or not the preview window is to have a control (system menu) box in the upper left hand corner when the report is printed to a window.

Usage

```
[form.]Report.WindowControlBox[ := {True|False} ];
```

For example:

```
Crp1.WindowControlBox := True;
```

«Specifies that a control box (system menu) is to appear in the preview window.»

Remarks

- Assign True if you want the window to contain a control box. Select False if you do not.
- Specify a value here only if you are printing to a window, if *Destination*, *Page 744*, is set to *toWindow*.

Data Type

Boolean

Availability

Design time; Runtime

WindowControls

Description

Specifies whether or not the print controls are to appear in the preview window when printing a report to a window.

Usage

```
[form.]Report.WindowControls[ :={True|False}];
```

For example:

```
Crpel.WindowControls := True;
```

«Specifies that print controls are to appear in the preview window.»

Data Type

Boolean

Availability

Design time; Runtime

WindowHeight

Description

Sets the height of the preview window when the report is printed to a window.

Usagepreview window

```
[form.]Report.WindowHeight[ :=Height];
```

For example:

```
Crpel.WindowHeight := 300;
```

«Sets the height of the preview window to 300 grasps.»

Remarks

- If you are not satisfied with the default settings, enter the external height you want for your preview window in pixels.
- Select a value here only if you are printing to a window, if *Destination, Page 744*, is set to toWindow.

Data Type

TWindowPosition

Availability

Design time; Runtime

WindowLeft

Description

Sets the distance, in pixels, that the preview window is to appear from the left edge of the parent window. If the preview window is a top level window, then the distance is measured from the left edge of the screen.

Usage

```
[form.]Report.WindowLeft[:=Distance];
```

For example:

```
Crpel.WindowLeft := 100;
```

«Sets the left edge of the preview window 100 twips from the left edge of the screen.»

Remarks

- If you are not satisfied with the default settings, enter the number of pixels you want between the left edge of the screen and the left edge of your window.
- Specify a value here only if you are printing to a window, if *Destination, Page 744*, is set to toWindow.

Data Type

Word

Availability

Design time; Runtime

WindowMaxButton

Description

Specifies whether or not the preview window is to have a maximize button when the report is printed to a window.

Usage

```
[form.]Report.WindowMaxButton[ := {True|False}] ;
```

For example:

```
Crpe1.WindowMaxButton := False;
```

«Specifies that no Maximize button is to appear in the preview window.»

Remarks

- Specify True if you want the window to contain a maximize button. Select False if you do not.
- Specify a value here only if you are printing to a window, if *Destination*, *Page 744*, is set to toWindow.

Data Type

TWindowPosition

Availability

Design time; Runtime

WindowMinButton

Description

Specifies whether or not the preview window is to have a minimize button when the report is printed to a window.

Usage

```
[form.]Report.WindowMinButton[ := {True|False} ];
```

For example:

```
Crpel.WindowMinButton := True;
```

«Specifies that a Minimize button is to appear in the preview window.»

Remarks

- Specify True if you want the window to contain a minimize button. Select False if you do not.
- Specify a value here only if you are printing to a window, if *Destination*, Page 744, is set to toWindow.

Data Type

Boolean

Availability

Design time; Runtime

WindowParentHandle

Description

Specifies the window handle for the parent window to be used for this preview window (except when the preview window is to be a top level window, in which case this parameter is 0). The property is for people who want their report to print to a window control (i.e., a panel).

Usage

```
[form.]Report.WindowParentHandle[ := handle];
```

For example:

```
crpel.windowparenthandle := form1.panel1.handle;
```

«Prints the report to panel1 inside of form1.»

Data Type

Integer

Availability

Runtime

WindowState

Description

Sets the state of the preview window, normal, minimized, or maximized, when the report is printed to a preview window.

Usage

```
[form.]Report.WindowState[ :=TWindowState];
```

For example:

```
Crp1.WindowState := wsMaximized;
```

«When the report is printed to a preview window, the preview window appears maximized when opened.»

Remarks

Use the following values to set the WindowState property:

- **wsNormal**

The preview window appears neither minimized nor maximized. It appears in a default size and position previously defined by your application or by Windows.

- **wsMinimized**

The preview window appears minimized as an icon close to the lower left hand corner of the screen. The icon can be restored to display the window in a normal state.

- **wsMaximized**

The preview window is maximized when opened to fill the entire screen.

Data Type

TWindowState = (wsNormal, wsMinimized, wsMaximized);

Availability

Design time; Runtime

WindowTitle

Description

Specifies the title you want to appear in the preview window title bar when the report is printed to a window.

Usage

```
[form.]Report.WindowTitle[:=Title];
```

For example:

```
Crpel.WindowTitle := 'Quarterly Earnings';
```

«Sets the title of the preview window (the string that appears on the title bar) to “Quarterly Earnings”.»

Remarks

- Make sure that the title is enclosed in quotes.
- Select a value here only if you are printing to a window, if *Destination*, *Page 744*, is set to *toWindow*.

Data Type

String

Availability

Design time; Runtime

WindowTop

Description

Sets the distance, in pixels, that the preview window is to appear from the top edge of the parent window. If the preview window is a top level window, then the distance is measured from the top edge of the screen.

Usage

```
[form.]Report.WindowTop[:=Distance];
```

For example:

```
Crpe1.WindowTop := 100;
```

«Sets the top edge of the preview window 100 pixels from the top of the screen.»

Remarks

- If you are not satisfied with the default setting, enter the number of pixels you want between the top of the screen and the top of your window.
- Select a value here only if you are printing to a window, if *Destination, Page 744*, is set to toWindow.

Data Type

TWindowPosition

Availability

Design time; Runtime

WindowWidth

Description

Specifies the width of the preview window in pixels.

Usage

```
[form.]Report.WindowWidth[ :=Width];
```

For example:

```
Crpe1.WindowWidth := 480;
```

«Specifies a preview window 480 twips wide.»

Remarks

- If you are not satisfied with the default setting, enter the external width of your window, in pixels.
- Select a value here only if you are printing to a window, if *Destination, Page 744*, is set to toWindow.

Data Type

TWindowPosition

Availability

Design time; Runtime

ZoomMagnification

Description

Sets the magnification factor for the report in the preview window to a value from 25% to 400% of the actual size.

Usage

```
[form.]Report.ZoomMagnification := [PercentZoomLevel];
```

For example:

```
Crpe1.ZoomMagnification := 150;
```

«Sets the magnification/zoom level to 150% for the current report.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after *Action*, Page 740, or *Execute*, Page 753.

Availability

Runtime

ZoomPreviewWindow

Description

Changes the magnification of the preview window to a specified level.

Usage

```
[form.]Report.ZoomPreviewWindow [:=TPreviewWindowZoom];
```

For example:

```
Crpel.ZoomPreviewWindow := pwFullScreen;
```

«Zooms the preview window to full size.»

Data Type

TPreviewWindowZoom = (pwFullScreen, pwFitOneSize, pwFitBothSides);

Availability

Runtime

METHODS

ChangeTableLocation

Description

This method is used to change the location of a table used in a report.

Usage

```
[form.]Report.ChangeTableLocation (TableArrayIndex,TableLocationInfo);
```

For example:

```
var
  tablLocInfo: PETableLocation;
  newLocation: PChar;
  NthTable: smallint;
begin
  NthTable := 3;
  newLocation := 'C:\CRW\CRAZE.MDB';
  tablLocInfo.StructSize := PE_SIZEOF_TABLE_LOCATION;
  tableLocInfo.Location := newLocation;
  Crpel.ChangeTableLocation (NthTable, tablelocInfo);
end;
```

Availability

Runtime

Clear

Description

This method is used to clear string properties and array properties.

Usage

```
[form.]Report.[StringProperty].Clear;
```

For example:

```
Crpel.SelectionFormula.Clear;
```

«This code reinitializes the array of any strings that were previously set.»

Remarks

Resets the VCL if you wish to use the VCL for more than one report.

For example:

```
Crpel.Clear;
```

Availability

Runtime

CloseWindow

Description

This method is used to close the associated preview window.

Usage

```
[form.]Report.CloseWindow;
```

For example:

```
Crpel.CloseWindow;
```

«Closes the current report that is displayed.»

Remarks

Closes the current preview window. This method is used mostly when not using standard window controls.

Availability

Runtime

ExportPrintWindow

Description

Exports the report displayed in the preview window. It will bring up the Export dialog boxes for the user to fill in the required information.

Usage

```
[form.]Report.ExportPrintWindow([{True|False}, {True|False}]);
```

For example:

```
Crpe1.ExportPrintWindow(True,True);
```

«Exports the report that is currently displayed in the window.»

Remarks

The first Boolean parameter is to default the method to ExportToMail and the second is WaitUntilDone and should always be set to true.

Availability

Runtime

FetchSelectionFormula

Description

Returns the selection formula from the current report.

Usage

```
[form.]Report.FetchSelectionFormula;
```

For example:

```
SelectionFormula := Crpel.FetchSelectionFormula;
```

Remarks

This method does not populate *SelectionFormula*, *Page 808*, and it does not conflict with setting the property.

Focused

Description

This method returns True if the associated preview window has the input focus.

Usage

```
[form.]Report.Focused;
```

For example:

```
If Crpel.Focused then  
ShowMessage ('Preview has focus');
```

«Prints the message, “Preview has focus” if the report is in the forefront with control.»

Availability

Runtime

LogoffServer

Description

The LogoffServer method terminates the specified database connection established earlier with the *LogonServer*, *Page 828* method.

Usage

```
[form.]Report.LogoffServer (connectionID);
```

For example:

```
Result := Crpel.LogoffServer (1);
```

«Terminates database connection 1 and only that connection. Returns successful/unsuccessful.»

Parameters

Parameter	Description
connectionId	Integer value that specifies a specific database connection established with <i>LogonServer</i> , <i>Page 828</i> .

LogonServer

Description

The LogonServer method logs on to the specified server and returns a unique connection ID which can be used to log off of this server using *LogoffServer*, *Page 827*.

Usage

```
[form.]Report.LogonServer ('dllName', 'ServerName', 'DatabaseName',  
'UserID', 'Password');
```

For example:

```
connectionId = Crpel.LogonServer ('pdsodbc.dll', 'Accounting', 'bobg',  
'Administration', 'bigboard');
```

«Connects to the “Administration” database via the “Accounting” datasource using the user ID “bobg” and the password “bigboard”.»

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to, for example, “PDSODBC.DLL”. Note that the dllName must be enclosed in quotes. DLL names have the following naming convention:

<i>Convention</i>	<i>Description</i>
PDB*.DLL	For standard (non-SQL) databases.
PDS*.DLL	For SQL/ODBC databases.

<i>Parameter</i>	<i>Description</i>
ServerName	Specifies the logon name for the server used to create the report. For ODBC, use the data source name.
DatabaseName	Specifies the logon name for the database used to create the report.
UserID	Specifies the user ID necessary to log on to the server.
Password	Specifies the password necessary to log on to the server. When you are using this structure to retrieve information using the PEGetNthTableLogOnInfo, the password parameter is undefined.

PageCount

Description

Returns the number of pages in the report.

Usage

```
[form.]Report.PageCount;
```

For example:

```
NumberOfPages := Crp1.PageCount;
```

Remarks

PageCount must be called after *Action*, *Page 740*, or *Execute*, *Page 753*, to get a valid page count.

PrintWindow

Description

Prints the report displayed in the preview window to the printer.

Usage

```
[form.]Report.PrintWindow([{True|False}]);
```

For example:

```
Crpe1.PrintWindow(True);
```

Remarks

Always set the WaitUntilDone parameter to True.

Availability

Runtime

ReplaceSelectionFormula

Description

Overrides the selection formula from in the current report with the string that is passed.

Usage

```
[form.]Report.ReplaceSelectionFormula ('SelectionFormulaString');
```

For example:

```
Crpe1.ReplaceSelectionFormula ('{company.State}="CA"');
```

Remarks

An exception will be raised if this method is called after setting *SelectionFormula*, *Page 808*. Setting the SelectionFormula property directly will always override the use of the ReplaceSelectionFormula method.

ReportWindowHandle

Description

Returns a handle to the associated preview window.

Usage

```
[form.]Report.ReportWindowHandle;
```

For example:

```
Type  
  MyWin : hWnd;  
  
Begin  
  Crpel.Execute;  
  MyWin := Crpel.ReportWindowHandle;  
  While IsWindow(MyWin) Do  
    Application.ProcessMessages;  
End;
```

«This sample will halt the execution of the application until the preview window is closed.»

Availability

Runtime

RetrieveDataFiles

Description

Retrieves the total number of tables for the current report.

Usage

```
[form.]Report.RetrieveDataFiles;
```

For example:

```
NumberofDatafiles: Crpel.RetrieveDatafiles;
```

Remarks

This integer can be used to examine the contents of *RetrieveTableData*, *Page 802*.

Availability

Runtime

RetrieveLogOnInfo

Description

Retrieves logon information (except password) for all “tables” in the current report, populates *LogOnInfo*, *Page 767*, and returns the number of “tables” in the report.

Usage

```
[form.]Report.RetrieveLogOnInfo;
```

For example:

```
NumberofTables := crpel.RetrieveLogOnInfo;
```

RetrieveSQLQuery

Description

Retrieves SQL Query from the current report and populates the SQLQuery property.

Usage

```
[form.]Report.RetrieveSQLQuery;
```

For example:

```
Result := crpel.RetrieveSQLQuery;
```

Remarks

Can be used with the SQLQuery property to obtain the SQL statement pertaining to the report.

RetrieveStoredProcParams

Description

Retrieves all Stored Procedure Parameters from the current report, populates *StoredProcParam*, *Page 813*, and returns the number of parameters.

Usage

```
[form.]Report.RetrieveStoredProcParams;
```

For example:

```
NumberOfParams := crpel.RetrieveStoredProcParams;
```

Remarks

Can be used with the StoredProcParams to examine the parameters of the stored procedure. This method can only be called AFTER ReportFileName has been set.

Related Report Engine Functions

PEGetNParams, Page 184

PEGetNthParam, Page 195

SetFocus

Description

The SetFocus method gives the input focus to the associated preview window.

Usage

```
[form.]Report.SetFocus;
```

For example:

```
If Not Crpel.Focused then  
  Crpel.SetFocus;
```

Availability

Runtime

ShowFirstPage

Description

The ShowFirstPage method will display the first page in the associated preview window.

Usage

```
[form.]Report.ShowFirstPage;
```

For example:

```
Crpe1.ShowFirstPage;
```

Availability

Runtime

ShowLastPage

Description

The ShowLastPage method will display the last page in the associated preview window.

Usage

```
[form.]Report.ShowLastPage;
```

For example:

```
Crpe1.ShowLastPage;
```

Availability

Runtime

ShowNextPage

Description

The ShowNextPage method will display the next page in the associated preview window.

Usage

```
[form.]Report.ShowNextPage;
```

For example:

```
Crpe1.ShowNextPage;
```

Remarks

Due to changes in the Report Engine, in versions 1.08.21 and 2.00.13.32 and earlier of the Crystal VCL this method will only be enabled after the user has previewed to the last page of the report.

Availability

Runtime

ShowPage

Description

Displays a specific page of the report in the preview window.

Usage

```
[form.]Report.ShowPage [(PageToShow)];
```

For example:

```
Crpe1.PageShow(3);
```

«Shows the third page of the report.»

Remarks

- This method is only valid when the report is printed to a preview window.
- This method must be called after the *Action*, *Page 740*, or *Execute*, *Page 753*.

ShowPreviousPage

Description

The ShowPreviousPage method will display the previous page in the associated preview window.

Usage

```
[form.]Report.ShowPreviousPage;
```

For example:

```
Crpe1.ShowPreviousPage;
```

Availability

Runtime

EVENTS

OnExecute

Description

Called when the control receives either *Execute*, [Page 753](#), or *Action*, [Page 740](#).

Usage

If you double-click on this from the Events page, this procedure will be created:

```
procedure TForm1.CrpelExecute(Sender: TObject; var Cancel: Boolean);  
begin  
end;
```

Remarks

- This procedure will be executed when Action:=1 is set or Execute is called.
- Sender = standard Delphi sending object.
- Cancel = by setting to true, will cancel printing of report.

Availability

Runtime

OnLoadDataFiles

Description

Called when the current job is open, but not started.

Usage

If you double-click on this from the Events page, this procedure will be created:

```
procedure TForm1.CrpelLoadDataFiles(Sender: TObject; const Count: Integer;
var Cancel: Boolean);
begin
end;
```

Remarks

- This procedure will be executed when *Action*, *Page 740*, is set to 1 or *Execute*, *Page 753*, is called.
 - Sender = standard Delphi sending object.
 - Count = the number of tables from the report.
 - Cancel = by setting to true, will cancel printing of report.
- The DataFilesLocation property will override any changes you make to the DataFiles property in this event.

Availability

Runtime

TUTORIALS

Resetting Array Properties

To clear an array property and release the memory associated with it, assign an empty string to the first element of the array.

For example, to clear all the strings from the DataFiles property:

```
Crpel.DataFiles[0] := EmptyStr;
```

Access Notes Sparse Array Properties

The Nth array element may be set without having to set up to the previous N-1 elements.

For example, the 7th table location may be set by issuing:

```
Crpel.DataFiles[6] := sTableName; on it's own.
```

Resetting Sort Array Properties

To clear a Sort Field array property and release the memory associated with it, assign an empty string to the first element of the array. This will also clear all the sort fields in your report.

For example, to clear all the strings from the SortFields property:

```
Crpe1.SortFields[0] := EmptyStr;
```

Resetting TCrpe String Properties

To clear a TCrpeString String List property and release the memory associated with it, call the property's Clear method.

For example, to clear all the strings from the SQLQuery property:

```
Crpe1.SQLQuery.Clear;
```

11

The Crystal Report Engine Class Library Reference

What you will find in this chapter...

CLASS LIBRARY SUPPORT STRUCTURES, Page 931

REPORT ENGINE CLASSES, Page 956

CRPEngine::CanClose

Syntax

```
BOOL CanClose();
```

Remarks

Checks to see whether or not the Crystal Report Engine is busy. Errors can occur in your application or on a system if you attempt to close the Crystal Report Engine while it is processing a print job. Use this method before attempting to close the Crystal Report Engine in an application (for example, when the user tries to exit) to determine if the Report Engine can be closed safely.

Return Value

TRUE if the Crystal Report Engine can be closed, FALSE if the Report Engine is still busy.

CRPEngine::Close

Syntax

```
void Close();
```

Remarks

This method closes the Crystal Report Engine.

Related Topics

PECloseEngine, Page 127

CRPEngine::CRPEngine

Syntax

```
CRPEngine(BOOL open);
```

Parameters

Parameter	Description
open	Indicates whether or not the Crystal Report Engine should be opened when the CRPEngine object is created.

Remarks

This is the constructor for the class. If open is true, the actual Seagate Crystal Reports DLL is opened (crpe32.dll). Print jobs may only be opened if the engine itself is open. If originally opened with open = FALSE, the engine may be opened later with the CRPEngine::Open method.

CRPEngine::GetEngine

Syntax

```
static CRPEngine *GetEngine();
```

Remarks

This method may be used to get a pointer to the current CRPEngine object being used in the application.

Return Value

A pointer to the current CRPEngine object.

CRPEngine::GetEngineStatus

Syntax

```
static Status GetEngineStatus();
```

Remarks

This method may be used to get the current engine status. Possible values returned are:

Value	Description
engineOpen	The Crystal Report Engine is currently open.
engineClosed	The Crystal Report Engine has been closed, or is not yet open.

<i>Value</i>	<i>Description</i>
engineMissing	No Crystal Report Engine is available. Make sure the Crystal Report Engine DLL is located in a directory that appears in your PATH.

Return Value

A value of the Status enumerated type. Indicates the current status of the Crystal Report Engine.

CRPEngine::GetErrorCode

Syntax

```
short GetErrorCode();
```

Remarks

This method returns the current error code of the Crystal Report Engine. When a call to another function fails, this call gets the error code that was generated so you can take some action based on that error code.

Return Value

The current Crystal Report Engine error code. 0 if no error has occurred.

Related Topics

PEGetErrorCode, Page 152

CRPEngine::GetErrorText

Syntax

```
CString GetErrorText();
```

Remarks

This method returns a descriptive Crystal Report Engine error message.

Return Value

A CString object containing the text for the current error.

Related Topics

PEGetErrorText, Page 153

CRPEngine::GetNPrintJobs

Syntax

```
int GetNPrintJobs ( );
```

Remarks

This method returns the number of print job (CRPEJob) objects that currently exist for the current CRPEngine object.

Return Value

The number of CRPEJob objects currently open. Returns -1 if an error occurs

CRPEngine::GetVersion

Syntax

```
short GetVersion(short versionRequested);
```

Parameters

Parameter	Description
versionRequested	Specifies whether the version number of the Crystal Report Engine or the Crystal Report Engine DLL is being requested. Possible values are:

Value	Meaning
PEP_GV_DLL	Returns the version of the DLL.

<i>Value</i>	<i>Meaning</i>
PEP_GV_ENGINE	Returns the version of the Crystal Report Engine.

Remarks

This method returns the version number of the Crystal Report Engine DLL (CRPE32.DLL) or the version of the Crystal Report Engine itself. The high-order byte is the major version number and the low-order byte is the minor version number.

This method can be used whenever you build functionality into a report that may not be available in earlier versions of the Crystal Report Engine and you need to verify the version number first. The function can be a safeguard for users who have more than one version of the Crystal Report Engine with the older version earlier in the path than the new version.

Return Value

The version number of the currently used Crystal Report Engine or Crystal Report Engine DLL.

Related Properties

PEGetVersion, Page 231

CRPEngine::LogOffServer

Syntax

```
BOOL LogOffServer(const char *dllName, const CRPELogOnInfo *logOnInfo);
```

Parameters

<i>Parameter</i>	<i>Description</i>
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to.
logOnInfo	Specifies a pointer to a CRPELogOnInfo structure.

Remarks

This method closes an existing connection to an SQL server. Connection information is provided through the *CRPELogOnInfo::CRPELogOnInfo, Page 943*, structure.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PLogOffServer, Page 240

CRPEngine::LogOnServer

Syntax

```
BOOL LogOnServer(const char *dllName, const CRPELogOnInfo *logOnInfo);
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to.
logOnInfo	Specifies a pointer to a CRPELogOnInfo structure.

Remarks

This method opens a connection to an SQL server. More than one print job may use this connection to an SQL server. Information required to establish a connection is provided through the *CRPELogOnInfo::CRPELogOnInfo, Page 943*, structure.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PLogOnServer, Page 241

CRPEngine::LogOnSQLServerWithPrivateInfo

Syntax

```
BOOL LogOnSQLServerWithPrivateInfo (const char *dllName, void  
*privateInfo);
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL that was used when establishing a connection to the Server when the report was first created. For example, if a report was created using an ODBC data source, specify the filename "PDSODBC.DLL". For more information on possible database DLLs, refer to Runtime File Requirements online Help.
privateInfo	Specifies the application's handle to the Server connection. In your application, a connection to the Server must already be established before this method is called. Pass the HDBC (handle to a database connection) from this connection to the privateInfo parameter.

Remarks

Use this class method to allow the Crystal Report Engine to “piggyback” your application’s existing connection to a Server. This lowers the number of connections established by a workstation, reducing application time and network traffic. It also prevents a Seagate Crystal Reports Log Off call from disconnecting an application’s existing connection to the Server.

The CRPEngine::LogOnSQLServerWithPrivateInfo method can only be used with database connections established by ODBC or Q+E Library 2.0. Any other database connections can not be accessed by this method.

To obtain an HDBC for an ODBC connection, use the following function calls (see the ODBC SDK 2.0 manual for more information):

Value	Meaning
SQLAllocEnv	Initializes the ODBC call level interface and allocates memory for an environment handle.
SQLAllocConnect	Returns an ODBC HDBC.

To obtain an HDBC for a Q+E Library connection, use the following function calls (see the InterSolv DataDirect Developer's Toolkit for more information):

<i>Value</i>	<i>Meaning</i>
qeConnect	Opens a connection to the server.
qeGetODBCHdbc	Returns the ODBC HDBC.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEngine::Open

Syntax

```
BOOL Open();
```

Remarks

This method opens the Crystal Report Engine. This method is only necessary if you constructed the CRPEngine object with the *open* parameter equal to FALSE. If you set the *open* parameter to TRUE when you constructed the CRPEngine object, this method is unnecessary.

Return Value

Returns TRUE if the Crystal Report Engine was opened successfully, FALSE if something went wrong.

Related Topics

CRPEngine::CRPEngine, Page 840

PEOpenEngine, Page 247

CRPEngine::OpenJob

Syntax

```
CRPEJob *OpenJob(const char *reportFileName);
```

Parameters

Parameter	Description
reportFileName	The name and path (if necessary) of the report file being opened for the specified print job.

Remarks

This method opens the report specified by reportFileName. A pointer to a CRPEJob object is returned. Print job attributes may be referenced through this object.

Return Value

A pointer to a CRPEJob object for the opened print job.

Related Properties

PEOpenPrintJob, Page 248

CRPEngine::PrintReport

Syntax

```
short PrintReport(const char *reportFilePath, BOOL toPrinter, BOOL  
toWindow, const char *title, int left, int top, int width, int height,  
DWORD style, CWnd *parentWindow);
```

Parameters

Parameter	Description
reportFilePath	Specifies the name and path of the report to be printed.
toPrinter	Specifies whether or not the report is to be sent to the default printer.
toWindow	Specifies whether or not the report is to be displayed in a preview window.
title	Specifies the title you want to appear in the title bar if the report is being sent to a preview window.
left	Specifies the x coordinate of the upper left hand corner of the preview window, in device coordinates.
top	Specifies the y coordinate of the upper left hand corner of the preview window, in device coordinates.

<i>Parameter</i>	<i>Description</i>
width	Specifies the width of the preview window, in device coordinates.
height	Specifies the height of the preview window, in device coordinates.
style	Specifies the style of the window being created. Style setting can be combined using the bitwise “OR” operator. Refer to the CWnd class in the Microsoft Foundation Class Library reference for possible window styles.
parentWindow	Specifies a pointer to the CWnd object for the window that is the parent of the preview window. Specify NULL if the preview window will not have a parent window.

Remarks

This method provides a quick but limited way to print a report. The report may only be output to a printer or preview window. Use this class method any time you simply want to print a report from an application without giving the user the ability to customize the report.

Return Value

0 if the call is successful. Returns an error code if something goes wrong.

Related Topics

PEPrintReport, Page 263

class CRPEJob: public CObject

In order to open a particular report it is first necessary to have an open Crystal Report Engine object in the application. You may then call the *CRPEngine::OpenJob, Page 847*, member function specifying the report file name to open. If successful, you will be returned a pointer to a CRPEJob object. It is through this object that you may modify the print job attributes as well as output the report in various formats. Almost all of these methods correspond to similar functions available in the Crystal Report Engine API.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEJob	Constructs a CPreview WINDOWRPEJob object. Used internally by <i>CRPEngine::OpenJob, Page 847</i> .

CRPEJob::Cancel

Syntax

```
void Cancel();
```

Remarks

This method cancels processing of a print job. It can be tied to a control that allows the user to cancel a print job in progress.

Related Topics

PECancelPrintJob, Page 121

CRPEJob::CheckFormula

Syntax

```
BOOL CheckFormula (const char *formulaName);
```

Parameters

Parameter	Description
formulaName	The name of the formula that you need to check for errors.

Remarks

Use this method to check a formula for validity. This method works just like the Check button in the Formula Editor. If the named formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is okay, FALSE if there is an error in the formula or something goes wrong.

Related Topics

PECheckFormula, Page 123

CRPEJob::CheckGroupSelectionFormula

Syntax

```
BOOL CheckGroupSelectionFormula () ;
```

Remarks

Use this method to check the group selection formula for the report for errors. This method works just like the Check button in the Formula Editor. If the group selection formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is okay, FALSE if there is an error in the formula or something goes wrong.

Related Topics

PECheckGroupSelectionFormula, Page 124

CRPEJob::CheckSelectionFormula

Syntax

```
BOOL Check SelectionFormula () ;
```

Remarks

Use this method to check the record selection formula for the report for errors. This method works just like the Check button in the Formula Editor. If the selection formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is okay, FALSE if there is an error in the formula or something goes wrong.

Related Topics

PECheckSelectionFormula, Page 126

CRPEJob::Close

Syntax

```
void Close();
```

Remarks

This method closes the print job. It also calls the destructor for the CRPEJob object. If printing has not yet finished when this method is called, it continues until the job is completely printed. If the preview window is open, it stays open.

Related Topics

PEClosePrintJob, Page 129

CRPEJob::CloseWindow

Syntax

```
void CloseWindow();
```

Remarks

This method closes the preview window. If you are customizing preview window controls, implement this method to allow the user to close the preview window when finished viewing the report.

Related Topics

PECloseWindow, Page 131

CRPEJob::CRPEJob

Syntax

```
CRPEJob(short jobHandle);
```

Parameters

Parameter	Description
jobHandle	Handle to the job created by CRPEngine::OpenJob().

Remarks

This is the constructor for the class. It also stores the job number internally for future use in Crystal Report Engine API calls.

NOTE: This class constructor is automatically called by CRPEngine::OpenJob(). Do not call this constructor from within your own code.

CRPEJob::DeleteNthGroupSortField

Syntax

```
BOOL DeleteNthGroupSortField (short sortFieldN);
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the group sort field you want to delete. The first group sort field added to the report is field 0, the second is 1, etc.

Remarks

This method deletes the group sort field at the specified position. The group and group summary are not removed from the report, but the field is removed from the list of group sort fields, and the summary data appearing in the group field is no longer sorted.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEDeleteNthGroupSortField, Page 136

CRPEJob::DeleteNthSortField

Syntax

```
BOOL DeleteNthSortField(short sortFieldN);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sortFieldN	Specifies the number of the sort field you want to delete. The first sort field added to the report is field 0, the second is 1, etc.

Remarks

This method deletes the specified sort field from the report. The field is not deleted from the report, but it is removed from the list of sort fields, and data in that field no longer appears sorted.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEDeleteNthSortField, Page 138

CRPEJob::DiscardSavedData

Syntax

```
BOOL DiscardSavedData ( );
```

Remarks

Discards data that was previously saved with the report. If a report has been saved with data, you can use this function to discard the saved data, forcing the Crystal Report Engine to retrieve new data when the report is printed.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEDiscardSavedData, Page 140

CRPEJob::EnableProgressDialog

Syntax

```
BOOL EnableProgressDialog (BOOL enable);
```

Parameters

Parameter	Description
enable	Specifies whether or not the progress dialog box is enabled. If enable is set to TRUE (1), the progress dialog box is enabled. If it's set to FALSE (0), the dialog box is disabled.

Remarks

The EnableProgressDialog function enables/disables the display of the Progress dialog box. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEEnableProgressDialog, Page 142

CRPEJob::ExportPrintWindow

Syntax

```
BOOL ExportPrintWindow(BOOL toMail);
```

Parameters

Parameter	Description
toMail	Specifies whether or not the report file should be exported to an e-mail address. If TRUE, the file is exported to e-mail. If FALSE, the file is exported to a disk file.

Remarks

This method exports the report displayed in the preview window to a disk file or e-mail address. If you are customizing preview window controls, use this method to enable the user to preview the report in the preview window, and if everything looks satisfactory, to export the report to a disk file or e-mail address (in response to a user event - button click, menu command, etc.).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEJob::ExportTo

Syntax

```
BOOL ExportTo (const CRPEExportOptions *options);
```

Parameters

Parameter	Description
options	A pointer to a CRPEExportOptions structure.

Remarks

This method sets the output of the print job to be exported. The export format is specified through the options parameter. This method does not export the report, but specifies that when the report is printed, it will be exported to a disk file or e-mail address according to settings in the options parameter. To actually export the report, use *CRPEJob::Start*, *Page 928*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEExportTo, *Page 146*

CRPEJob::GetErrorCode

Syntax

```
short GetErrorCode ();
```

Remarks

This method gets the error code for the print job. When a call to another function fails, this call gets the error code that was generated so you can take some action based on that error code.

Return Value

A Seagate Crystal Reports Class Library error code. Return 0 (PEP_NOERROR) if no error has occurred.

Related Topics

PEGetErrorCode, Page 152

CRPEJob::GetErrorText

Syntax

```
CString GetErrorText ();
```

Remarks

This method returns a descriptive error message for the print job.

Return Value

A text description of the current Seagate Crystal Reports Class Library error if an error has occurred.

Related Topics

PEGetErrorText, Page 153

CRPEJob::GetFormula

Syntax

```
BOOL GetFormula(const char *formulaName, CString &formulaText);
```

Parameters

Parameter	Description
formulaName	The name of the formula for which you want to retrieve the formula string.
formulaText	The CString object passed here is loaded with the specified formula text when CRPEJob::GetFormula completes successfully.

Remarks

This method returns the formula text for the specified formula. CRPEJob::GetFormula is often used with *CRPEJob::SetFormula*, *Page 903*, to identify and then change an existing formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if the named formula does not exist in the report.

Related Topics

PEGGetFormula, *Page 156*

CRPEJob::GetGraphData

Syntax

```
BOOL GetGraphData (short sectionCode, short graphN, CRPEGraphDataInfo  
*graphDataInfo);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to get information from. Possible values are:
<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to get graph data information from. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphDataInfo	A pointer to a CRPEGraphDataInfostructure.

Remarks

Use this method to find out what data in your report is being used by a specified graph when the graph is created. This information includes which groups are used to create the rows and columns of the graph and which summary field in the report is used to set the values of the risers in the graph.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetGraphData, Page 158

CRPEJob::GetGraphOptions

Syntax

```
BOOL GetGraphOptions (short sectionCode, short graphN, CRPEGraphOptions  
*graphOptions);
```

Parameters

Parameter	Description
<i>sectionCode</i>	Specifies the section of the report containing the graph you want to get graph display options for. Possible values are:

Value	Meaning
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

Parameter	Description
<i>graphN</i>	Specifies which graph within the section you want to get graph display options for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
<i>graphOptions</i>	A pointer to a CRPEGraphOptions structure.

Remarks

Use this method to obtain information about any of several graph options. These options include the minimum and maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetGraphOptions, Page 159

CRPEJob::GetGraphText

Syntax

```
BOOL GetGraphText (short sectionCode, short graphN, CRPEGraphTextInfo  
*graphTextInfo);
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to get title text for. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to get title text for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphTextInfo	A pointer to a CRPEGraphTextInfo structure.

Remarks

This method allows you to find out what text appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGGetGraphText, Page 161

CRPEJob::GetGraphType

Syntax

```
BOOL GetGraphType (short sectionCode, short graphN, short *graphType);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to find out the type of. Possible values are:
<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.

<i>Value</i>	<i>Meaning</i>
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to know the type of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphType	Retrieves a value indicating the type of graph that the specified graph is. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_SIDE_BY_SIDE_BAR_GRAPH	Side By Side bar graph
PEP_STACKED_BAR_GRAPH	Stacked bar graph
PEP_PERCENT_BAR_GRAPH	Percent bar graph
PEP_FADED_3D_SIDE_BY_SIDE_BAR_GRAPH	3D Side By Side bar graph
PEP_FADED_3D_STACKED_BAR_GRAPH	3D Stacked bar graph
PEP_FADED_3D_PERCENT_BAR_GRAPH	3D Percent bar graph
PEP_PIE_GRAPH	Pie graph
PEP_MULTIPLE_PIE_GRAPH	Multiple Pie graph
PEP_PROPORIONAL_MULTI_PIE_GRAPH	Weighted Pie graph
PEP_LINE_GRAPH	Line graph
PEP_AREA_GRAPH	Area graph
PEP_THREED_BAR_GRAPH	3D bar graph
PEP_USER_DEFINED_GRAPH	User Defined graph type

<i>Value</i>	<i>Meaning</i>
PEP_UNKNOWN_TYPE_GRAPH	Unknown graph type

Remarks

Identifies one of the available graph/chart types used for the specified graph in the specified section. Use this method to find out what type of graph is being displayed in the report. There are many types of graphs and charts possible with Seagate Crystal Reports. This method will return the type of graph or chart being displayed.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetGraphType, Page 162

CRPEJob::GetGroupCondition

Syntax

```
BOOL GetGroupCondition (short sectionCode, CString &conditionField,
short *condition, short *sortDirection);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the report section for which you want to get the grouping condition. Possible values are:

	<i>Value</i>	<i>Meaning</i>
	PEP_GROUPHEADER	Sets the group condition for the Group Header section.
	PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.

<i>Parameter</i>	<i>Description</i>
conditionField	Specifies the name of the group summary field for which you want to get the grouping condition.

<i>Parameter</i>	<i>Description</i>						
condition	<p>Retrieves the type of field being used as the condition field and the condition that creates a new group. Use one of the following masks to separate the condition type from the group condition:</p> <table border="1"> <thead> <tr> <th><i>Value</i></th><th><i>Meaning</i></th></tr> </thead> <tbody> <tr> <td>PEP_GC_CONDITIONMASK</td><td>Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain the group condition value.</td></tr> <tr> <td>PEP_GC_TYPEMASK</td><td>Obtains the type of field used for the group condition. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain a value representing the type of field used by the group condition.</td></tr> </tbody> </table>	<i>Value</i>	<i>Meaning</i>	PEP_GC_CONDITIONMASK	Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain the group condition value.	PEP_GC_TYPEMASK	Obtains the type of field used for the group condition. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain a value representing the type of field used by the group condition.
<i>Value</i>	<i>Meaning</i>						
PEP_GC_CONDITIONMASK	Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain the group condition value.						
PEP_GC_TYPEMASK	Obtains the type of field used for the group condition. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain a value representing the type of field used by the group condition.						

For group condition field types other than Date and Boolean, the group condition value of the *condition* parameter is PEP_GC_ANYCHANGE. For a group condition field of the type Date, the group condition value will be one of the following:

<i>Value</i>	<i>Meaning</i>
PEP_GC_DAILY	Triggers a grouping every time the date changes.
PEP_GC_WEEKLY	Triggers a grouping every time the date changes from one week to the next (a week runs from Sunday through Saturday).
PEP_GC_BIWEEKLY	Triggers a grouping every time the date changes from one two-week period to the next.
PEP_GC_SEMIMONTHLY	Triggers a grouping every time the date changes from one half-month period to the next.

<i>Value</i>	<i>Meaning</i>
PEP_GC_MONTHLY	Triggers a grouping every time the date changes from one month to the next.
PEP_GC_QUARTERLY	Triggers a grouping every time the date changes from one calendar quarter to the next.
PEP_GC_SEMIANNUALLY	Triggers a grouping every time the date changes from one half-year period to the next.
PEP_GC_ANNUALLY	Triggers a grouping every time the date changes from one year to the next.

For a group condition field of the type Boolean, the group condition value will be one of the following:

<i>Value</i>	<i>Meaning</i>
PEP_GC_TOYES	Triggers a grouping every time the sort and group by field changes from No to Yes.
PEP_GC_TONO	Triggers a grouping every time the sort and group by field changes from Yes to No.
PEP_GC_EVERYYES	Triggers a grouping every time the group and sort by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the group and sort by field value is No.
PEP_GC_NEXTISYES	Triggers a grouping every time the next value in the sort and group by field is Yes.
PEP_GC_NEXTISNO	Triggers a grouping every time the next value in the sort and group by field is No.

The group condition field type portion of the condition parameter will be one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_GC_TYPEOTHER	Any field type other than Date or Boolean. The group condition portion of the <i>condition</i> parameter will be PEP_GC_ANYCHANGE.
PEP_GC_TYPEDATE	A Date field is used to create the group summary field.
PEP_GC_TYPEBOOLEAN	A Boolean field is used to create the group summary field.

<i>Parameter</i>	<i>Description</i>
sortDirection	Obtains the sort direction for the group summary field. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts data in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts data in descending order (Z to A, 9 to 1).

Remarks

Determines the group condition information for a selected group section in the specified report. Use this method to find out the group condition for a group section. Use *CRPEJob::SetGroupCondition*, [Page 909](#), to change the group condition once it is known.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetGroupCondition, [Page 164](#)

CRPEJob::GetGroupSelectionFormula

Syntax

```
BOOL GetGroupSelectionFormula(CString &formulaText);
```

Parameters

<i>Parameter</i>	<i>Description</i>
formulaText	Specifies the existing group selection formula for the report.

Remarks

This method returns the formula text for the group selection formula.

CRPEJob::GetGroupSelectionFormula is often used with *CRPEJob::SetGroupSelectionFormula*, [Page 912](#), to identify and then change an existing group selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGGetGroupSelectionFormula, [Page 168](#)

CRPEJob::GetJobHandle

Syntax

```
short GetJobHandle();
```

Remarks

This method returns the job handle for the print job.

Return Value

The print job handle for the CRPEJob object.

CRPEJob::GetJobStatus

Syntax

```
short GetJobStatus(CRPEJobInfo * jobStatus);
```

Parameters

Parameter	Description
jobStatus	A pointer to a CRPEJobInfo structure.

Remarks

This method gets the current status of the print job. You can use it in a number of programming situations, for example:

- to trigger error messages, (i.e., when a print job fails due to insufficient memory, insufficient disk space, etc.),
- to trigger screen displays (hourglass, series of graphics, etc.) that confirm to the user that work is in progress, or
- to find out whether a job was cancelled by the user after *CRPEJob::Start*, [Page 928](#), was called.

Return Value

Returns one of the following values according to the status of the current print job:

Value	Description
-1	The Crystal Report Engine has not been opened, or a print job has not been established.
PEP_JOBNOTSTARTED	The job has not been started.
PEP_JOBINPROGRESS	The job is currently in progress.
PEP_JOBCOMPLETED	The job has completed successfully.
PEP_JOBFAILED	The job has failed.
PEP_JOBCANCELLED	The job has been cancelled by the user.

Related Topics

PEGetJobStatus, [Page 171](#)

CRPEJob::GetMargins

Syntax

```
BOOL GetMargins (short *left, short *right, short *top, short *bottom);
```

Parameters

Parameter	Description
left	Retrieves the current setting of the left margin in twips.
right	Retrieves the current setting of the right margin in twips.
top	Retrieves the current setting of the top margin in twips.
bottom	Retrieves the current setting of the bottom margin in twips.

Remarks

Retrieves the page margin settings for the specified report. Use this method to find out what the currently set margins for the report are. Use *CRPEJob::SetMargins*, Page 912, to change report margins.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetMargins, Page 173

CRPEJob::GetMinimumSectionHeight

Syntax

```
BOOL GetMinimumSectionHeight (short sectionCode, short *height);
```

Parameters

Parameter	Description
sectionCode	Specifies the code for the report section for which you want to retrieve the minimum height. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Gets the minimum section height for all sections.
PEP_HEADERSECTION	Gets the minimum section height for the Page Header section.
PEP_GROUPHEADER	Gets the minimum section height for the Group Header section.
PEP_DETAILSECTION	Gets the minimum section height for the Details section.
PEP_GROUPFOOTER	Gets the minimum section height for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the minimum section height for the Grand Total section.
PEP_FOOTERSECTION	Gets the minimum section height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
height	Retrieves the minimum height, in twips, for the specified report section.

Remarks

Retrieves minimum section height information for selected sections in the specified report. Use this method to fetch the minimum section height and pass back using *CRPEJob::SetMinimumSectionHeight*, *Page 913*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetMinimumSectionHeight, *Page 175*

CRPEJob::GetNDetailCopies

Syntax

```
BOOL GetNDetailCopies (short *nCopies);
```

Parameters

<i>Parameter</i>	<i>Description</i>
nCopies	Retrieves the current setting for the number of times the Details section of the report will be printed.

Remarks

Returns the number of copies of each Details section in the report that are to be printed. Use this method to find out how many times each Details section of the report will be printed. To change the number of times each Details section is printed, use *CRPEJob::SetNDetailCopies*, *Page 914*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNDetailCopies, *Page 176*

CRPEJob::GetNFormulas

Syntax

```
short GetNFormulas ();
```

Remarks

Use this function to fetch the number of formulas in the report. To fetch the formula by number, use *CRPEJob::GetNthFormula*, *Page 877*.

Return Value

The number of formulas in the report. If no formulas exist in the report, 0 is returned. If an error occurs, -1 is returned.

Related Topics

PEGetNFormulas, Page 177

CRPEJob::GetNGroups

Syntax

```
short GetNGroups ();
```

Remarks

Use this method to fetch the number of group sections in the report.

Return Value

The number of group sections in the report. Returns -1 if an error occurs.

Related Topics

PEGetNGroups, Page 179

CRPEJob::GetNGroupSortFields

Syntax

```
short GetNGroupSortFields ();
```

Remarks

Returns the number of group sort fields in the specified report. This method is typically used as one of a series: GetNGroupSortFields (called once), *CRPEJob::GetNthGroupSortField, Page 878*, (called as many times as needed to identify the correct group sort field), and *CRPEJob::SetNthGroupSortField, Page 915*, (called once, when the correct group sort field is identified). The series can be used to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

Return Value

The number of group sort fields in the report. Returns 0 if there are no group sort fields. Returns -1 if an error occurs.

Related Topics

PEGetNGroupSortFields, Page 180

CRPEJob::GetNPages

Syntax

```
short GetNPages ();
```

Remarks

The GetNPages function retrieves the number of pages in the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNPages, Page 181

CRPEJob::GetNParameterFields

Syntax

```
short GetNParameterFields ();
```

Remarks

The GetNParameterFields function determines the number of parameter fields used in the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNParameterFields, Page 183

CRPEJob::GetNParams

Syntax

```
short GetNParams ();
```

Remarks

Use this method whenever you need to know how many parameters are required by a stored procedure in an SQL data base table. This method is usually used in conjunction with *CRPEJob::GetNthParam*, *Page 879*, or *CRPEJob::SetNthParam*, *Page 916*.

Return Value

The number of parameters in the current stored procedure being used to generate the report.

Related Topics

PEGetNParams, *Page 184*

CRPEJob::GetNSections

Syntax

```
short GetNSections ();
```

Remarks

The GetNSections function retrieves the number of sections in the specified report. By default, each report has five areas, each containing one section (Report Header, Page Header, Details, Report Footer, and Page Footer). Thus, if this function were applied to a default report, the return value would be five (5). As you add groups to your report or you add sections to one or more areas, the number of sections in the report increases.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNSections, *Page 185*

CRPEJob::GetNSortFields

Syntax

```
short GetNSortFields();
```

Remarks

This method gets the number of sort fields in the report.

Return Value

The number of sort fields defined in the report. Returns 0 (zero) if there are no sort fields in the report. Returns -1 if an error occurs.

Related Topics

PEGetNSortFields, Page 187

CRPEJob::GetNSubreportsInSection

Syntax

```
short GetNSubreportsInSection (short sectionCode);
```

Parameters

Parameter	Description
print job	Specifies the handle of the primary report from which you want to fetch information about the number of subreports in a section.
sectionCode	Specifies the section code of the section for which you want a subreport count. See <i>Working with section codes, Page 116</i> .

Remarks

The GetNSubreportsInSection function determines the number of subreports in the specified section.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNSubreportsInSection, Page 188

CRPEJob::GetNTables

Syntax

```
short GetNTables();
```

Remarks

This method returns the number of tables in the report.

Return Value

The number of tables used in the report. Returns -1 if an error occurs.

Related Topics

PEGetNTables, Page 188

CRPEJob::GetNthFormula

Syntax

```
BOOL GetNthFormula (short formulaN, CString &formulaName, CString  
&formulaText);
```

Parameters

Parameter	Description
formulaN	Specifies the number of the formula about which you want to gather information. The first formula added to your report is 0, the second is 1, etc.
formulaName	Retrieves the name of the formula specified.
formulaText	Retrieves the text of the formula specified.

Remarks

Use this function to obtain the formula name and formula text of a specific formula in the report. Once the formula name is obtained, formula text can be changed with *CRPEJob::SetFormula, Page 903*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthFormula, Page 191

CRPEJob::GetNthGroupSortField

Syntax

```
BOOL GetNthGroupSortField (short sortFieldN, CString &field, short  
*direction);
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the group sort field you want to retrieve. The first group sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1.
field	Retrieves the name of the group sort field.
direction	Retrieves a value indicating the sort direction. Possible values are:

Value	Meaning
PE_SF_ASCENDING	Sorted in Ascending order (A to Z, 1 to 9).
PE_SF_DESCENDING	Sorted in Descending order (Z to A, 9 to 1).

Remarks

Returns information about one of the group sort fields in the specified report: that is, it returns the name of the field and the direction (ascending or descending) of the sort. This method is typically used as one of a series: *CRPEJob::GetNGroupSortFields, Page 873*, (called once),

`CRPEJob::GetNthGroupSortField` (called as many times as needed to identify the correct group sort field), and *CRPEJob::SetNthGroupSortField, Page 915*, (called once when the correct sort field is identified). The series can be used to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthGroupSortField, Page 193

CRPEJob::GetNthParam

Syntax

```
BOOL GetNthParam (short paramN, CString &paramValue);
```

Parameters

Parameter	Description
paramN	Specifies which parameter in the stored procedure you want to get the value of. The first parameter of a stored procedure is 0, the second is 1, etc.
paramValue	Retrieves the current value of the specified parameter in the stored procedure.

Remarks

Gets the Nth parameter of a stored procedure. Use this method whenever you need to find out a particular parameter required by a stored procedure in a SQL database table.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNParams, Page 184

CRPEJob::GetNthParameterField

Syntax

```
BOOL GetNthParameterField (short parameterN, CRPEParameterFieldInfo  
*parameterInfo);
```

Parameters

<i>Parameter</i>	<i>Description</i>
<i>parameterN</i>	Specifies the number of the parameter field about which you want to retrieve information.
<i>parameterInfo</i>	A pointer to a CRPEParameterFieldInfo structure.

Remarks

Returns information about one of the parameter fields in the specified report; that is, it returns the name of the field, the data type, and information about the value set for the field. The name of the parameter field is returned as a string handle. This function is typically used as one of a series of functions: *CRPEJob::GetNParameterFields*, *Page 874*, (called once), *GetNthParameterField*, and *CRPEJob::SetNthParameterField*, *Page 917*, (called once when the correct parameter field is identified). The series can be used in a custom-print link to identify and then change an existing parameter field value at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthParameterField, *Page 197*

CRPEJob::GetNthParamInfo

Syntax

```
BOOL GetNthParamInfo (short paramN, CRPEParameterInfo *paramInfo);
```

Parameters

Parameter	Description
paramN	Specifies the number of the stored procedure parameter about which you want to gather information.
paramInfo	A pointer to a CRPEParameterInfo structure.

Remarks

The GetNthParamInfo function retrieves the name and type of a specified stored procedure parameter.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthParamInfo, Page 199

CRPEJob::GetNthSortField

Syntax

```
BOOL GetNthSortField(short sortFieldN, CString &field, short  
*direction);
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the sort field you want to retrieve. The first sort field added to the report is field 0, the second is 1, etc.
field	Assigned the name of the sort field if the call completes successfully.
direction	Assigned the sort direction of the sort field if the call completes successfully. The following values are possible:

Value	Meaning
PEP_SF_ASCENDING	Sorted in ascending order (A to Z, 1 to 9).

<i>Value</i>	<i>Meaning</i>
PEP_SF_DESCENDING	Sorted in descending order (Z to A, 9 to 1).

Remarks

This method gets the name of the sort field at the specified sort field position and the direction in which data is sorted (ascending or descending).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthSortField, Page 200

CRPEJob::GetNthSubreportInSection

Syntax

```
DWORD GetNthSubreportInSection ( short sectionCode, short subreportN );
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the report section that contains the subreport. See <i>Working with section codes, Page 116</i> .
subreportN	Specifies the number of the subreport in the specified section. subreportN is zero based. The first report in the section will be 0, the second will be 1, etc. If there are no subreports in the section, the function will return 0.

Remarks

The GetNthSubreportInSection function retrieves a handle that is required to retrieve the name of the subreport.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetNthSubreportInSection, Page 202

CRPEJob::GetNthTableLocation

Syntax

```
BOOL GetNthTableLocation(short tableN, CRPETableLocation  
*tableLocation);
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table for which you want to retrieve location information.
tableLocation	A pointer to a CRPETableLocation structure.

Remarks

This method gets table location information for the specified table in the report. This method is typically combined with the *CRPEJob::SetNthTableLocation, Page 919*, method to identify the location of a table and then to change it.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PEGetNthTableLocation, Page 203

CRPEJob::GetNthTableLogOnInfo

Syntax

```
BOOL GetNthTableLogonInfo(short tableN, CRPELogOnInfo *logonInfo);
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table from which you want to get log on information.
logonInfo	A pointer to a CRPELogOnInfo structure.

Remarks

This method gets SQL connection information for the specified table.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PEGetNthTableLogOnInfo, Page 205

CRPEJob::GetNthTableSessionInfo

Syntax

```
BOOL GetNthTableSessionInfo(short tableN, CRPESessionInfo * sessionInfo);
```

Parameters

Parameter	Description
tableN	A 0 indexed table number indicating which MS Access table in the report you wish to obtain the session information for.
sessionInfo	A pointer to a CRPESessionInfo structure.

Remarks

This methods gets session information for the specified Microsoft Access table. Many MS Access database tables require that a session be opened before the information in the table can be used. Use this method to obtain the session information (User ID and Session Handle) for a particular tab.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PEGetNthTableSessionInfo, Page 206

CRPEJob::GetNthTableType

Syntax

```
BOOL GetNthTableType( short  tableN, CRPETableType *tableType);
```

Parameters

Parameter	Description
tableN	0 indexed table number indicating which table in the report for which you want to determine the type.
tableType	A pointer to a CRPETableTypestructure.

Remarks

This method gets table type information for the specified table.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PEGetNthTableType, Page 207

CRPEJob::GetPrintDate

Syntax

```
BOOL GetPrintDate ( short *year, short *month, short *day);
```

Parameters

Parameter	Description
year	Retrieves the year for the current print date.
month	Retrieves the month for the current print date.

<i>Parameter</i>	<i>Description</i>
day	Retrieves the day for the current print date.

Remarks

Determines the print date (if any) that was specified with the report. Use this method to fetch the print date and pass back using *CRPEJob::SetPrintDate, Page 921*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetPrintDate, Page 209

CRPEJob::GetPrintOptions

Syntax

```
BOOL GetPrintOptions (CRPEPrintOptions *options);
```

Parameters

<i>Parameter</i>	<i>Description</i>
options	A pointer to a CRPEPrintOptions structure.

Remarks

Retrieves the print options specified for the report (the options that are set in the Print Setup common dialog box) and uses them to fill in the CRPEPrintOptions structure. Use this function to fetch print options from the report in order to update them and pass back using *CRPEJob::SetPrintOptions, Page 922*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetPrintOptions, Page 211

CRPEJob::GetReportTitle

Syntax

```
BOOL GetReportTitle (CString &title);
```

Parameters

Parameter	Description
title	Assigned the title of the report.

Remarks

This method gets the report title for the print job.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetReportTitle, Page 212

CRPEJob::GetSectionCode

Syntax

```
short GetSectionCode (short sectionN);
```

Parameters

Parameter	Description
printJob	Specifies the handle of the print job from which you want to identify a section code.
sectionN	Specifies the number of the section for which you want the section code. See <i>Working with section codes, Page 116</i> .

Remarks

The GetSectionCode function retrieves the section code for the specified section. A section code indicates the section type (Page Header, Details, and so forth). If there are multiple group sections it also identifies the group number, and if there are multiple sections in an area it identifies the section number.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetSectionCode, Page 215

CRPEJob::GetSectionFormat

Syntax

```
BOOL GetSectionFormat (short sectionCode, CRPESectionOptions *options);
```

Parameters

Parameter	Description
sectionCode	Specifies the code for the report section that you need to obtain format information for. Possible values are:
Value	Meaning
PEP_ALLSECTIONS	Gets the format information for all sections.
PEP_HEADERSECTION	Gets the format information for the Page Header section.
PEP_GROUPHEADER	Gets the format information for the Group Header section.
PEP_DETAILSECTION	Gets the format information for the Details section.
PEP_GROUPFOOTER	Gets the format information for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the format information for the Grand Total section.

<i>Value</i>	<i>Meaning</i>
PEP_FOOTERSECTION	Gets the format information for the Page Footer section.
<i>Parameter</i>	<i>Description</i>
options	A pointer to a CRPESectionOptions structure.

Remarks

Retrieves the section format settings for a selected section in the specified report and supplies the information by assigning values to the members of the CRPESectionOptions structure. Use this method in order to edit and update the section formats and pass information back using *CRPEJob::SetSectionFormat*, *Page 923*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetSectionFormat, *Page 217*

CRPEJob::GetSelectedPrinter

Syntax

```
BOOL GetSelectedPrinter (CString &driverName, CString &printerName,
CString &portName, DEVMODE **mode);
```

Parameters

<i>Parameter</i>	<i>Description</i>
driverName	Retrieves the name of the printer driver for the currently selected printer in the report.
printerName	Retrieves the name of the printer currently selected in the report.
portName	Retrieves the name of the port the currently selected printer is attached to. For example, "LPT1:".

<i>Parameter</i>	<i>Description</i>
mode	A DEVMODE structure that contains information on the currently selected printer, if the CRPEJob::GetSelectedPrinter method completes successfully. For more information on the DEVMODE structure, see <i>DEVMODE</i> , <i>Page 353</i> .

Remarks

If a non-default printer is specified in the report, this method retrieves information about that printer. If you need to know which printer has been specified with the report, use CRPEJob::GetSelectedPrinter to obtain the information. This can be useful to determine if the user has access to the printer specified in the report and to choose another printer if not.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetSelectedPrinter, *Page 222*

CRPEJob::GetSelectionFormula

Syntax

```
BOOL GetSelectionFormula(CString &formulaText);
```

Parameters

<i>Parameter</i>	<i>Description</i>
formulaText	Specifies the existing selection formula for the report.

Remarks

This method returns the formula text for the record selection formula.

CRPEJob::GetSelectionFormula is often used with CRPEJob::SetSelectionFormula, *Page 924*, to identify and then change an existing selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetSelectionFormula, Page 224

CRPEJob::GetSQLQuery

Syntax

```
BOOL GetSQLQuery (CString &query);
```

Parameters

Parameter	Description
query	Assigned the text of the SQL query being sent to the server if the call completes successfully.

Remarks

This method retrieves the SQL query that will be sent to the database server. This method can be used with *CRPEJob::SetSQLQuery, Page 925*, to retrieve and then change the SQL query for the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEGetSQLQuery, Page 226

CRPEJob::GetSubreportInfo

Syntax

```
BOOL GetSubreportInfo (DWORD subreportHandle, CRPESubreportInfo  
*subreportInfo);
```

Parameters

Parameter	Description
subreportHandle	Specifies the handle of the subreport about which you want to retrieve information.
subreportInfo	A pointer to a CRPESubreportInfo structure.

Remarks

The GetSubreportInfo function retrieves information about the specified subreport.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEJob::GetWindowHandle

Syntax

```
HWND GetWindowHandle();
```

Remarks

This method returns the Windows handle for the preview window. When this method returns successfully, you can use the HWND value with any Windows API function that requires a window handle as a parameter.

Return Value

The Windows Handle for the preview window.

Related Topics

PEGetWindowHandle, Page 232

CRPEJob::HasSavedData

Syntax

```
BOOL HasSavedData (BOOL *hasSavedData);
```

Parameters

Parameter	Description
printJob	Specifies the handle to the print job you want to query to find if it has saved data.
hasSavedData	Specifies a pointer to a memory address that indicates whether or not there is data saved with the report.

Remarks

The HasSavedData function queries a report to find if data is saved with it. With this information, you can determine whether or not the data needs to be refreshed before the report is printed.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEHasSavedData, Page 237

CRPEJob::IsJobFinished

Syntax

```
BOOL IsJobFinished();
```

Remarks

This method returns a Boolean value that indicates whether or not processing has finished for the print job. You can use this method any time you have a call that is contingent on a print job being finished.

Return Value

TRUE if processing has finished, FALSE if the job is in progress.

Related Topics

PEIsPrintJobFinished, Page 238

CRPEJob::NextWindowMagnification

Syntax

```
BOOL NextWindowMagnification();
```

Remarks

This method switches to the next preview Window magnification in order. Use this function to cycle through the three levels of preview window magnification whenever the report has been printed to a preview window. The three levels of magnification are: Full Page, Fit One Side, and Fit Both Sides.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PENextPrintWindowMagnification, Page 245

CRPEJob::OpenSubreportJob

Syntax

```
CRPEJob *OpenSubreportJob (const _TCHAR *subreportName);
```

Parameters

Parameter	Description
subreportName	Specifies the name of the subreport you want to access. This name is retrieved using CRPEJob::GetSubreportInfo.

Remarks

The OpenSubreportJob function opens the named subreport and returns a pointer to a CRPEJob object for the subreport. The CRPEJob object can be used to make subsequent changes to the subreport.

Return Value

A pointer to a class *CRPEJob: public CObject, Page 849*, object for the subreport.

Related Topics

PEOpenSubreport, Page 250

CRPEJob::OutputToPrinter

Syntax

```
BOOL OutputToPrinter (short nCopies = 1);
```

Parameters

Parameter	Description
nCopies	Specifies how many copies of the report are to be printed. Default is 1 copy.

Remarks

This method sets the output of the print job to the printer with the specified number of copies. This method does not print the report, but specifies that when the report is printed, it will be sent to a printer. To actually print the report, use *CRPEJob::Start, Page 928*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEOutputToPrinter, Page 254

CRPEJob::OutputToWindow

Syntax

```
BOOL OutputToWindow (const _TCHAR *title, int left, int top, int width,  
int height, int style, CWnd *parentWindow);  
  
BOOL OutputToWindow (const char *title, int left, int top, int width,  
int height, int style, CMDIFrameWnd *parentWindow);
```

Parameters

Parameter	Description
title	Specifies the title you want to appear in the title bar.
left	Specifies the x coordinate of the upper left hand corner of the preview window, in device coordinates.
top	Specifies the y coordinate of the upper left hand corner of the preview window, in device coordinates.
width	Specifies the width of the preview window, in device coordinates.
height	Specifies the height of the preview window, in device coordinates.
style	Specifies the style of the window being created. Style setting can be combined using the bitwise Or operator (). Refer to the CWnd class in the Microsoft Foundation Class Library reference for possible window styles.
parentWindow	Specifies a pointer to the CWnd object (in an SDI application) or the CMDIFrameWnd object (in an MDI application) for the window that is the parent of the preview window. Specify NULL if the preview window will not have a parent window.

Remarks

- This method sets the output of the print job to the preview window which will have the specified attributes. This method does not print the report, but specifies that when the report is printed, it will appear in a preview window. To actually print the report, use *CRPEJob::Start*, Page 928.
- This member function is overloaded to handle both SDI and MDI applications.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEOutputToPrinter, Page 254

CRPEJob::PrintControlsShowing

Syntax

```
BOOL PrintControlsShowing (BOOL *controlsShowing);
```

Parameters

Parameter	Description
controlsShowing	Returns a pointer to a TRUE value if the print controls will be shown in the preview window, FALSE if they will be hidden.

Remarks

Checks if the print controls are displayed in the preview window. Use *CRPEJob::ShowPrintControls*, [Page 928](#), to change whether or not print controls will appear in the preview window.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEPrintControlsShowing, [Page 261](#)

CRPEJob::PrintWindow

Syntax

```
BOOL PrintWindow( );
```

Remarks

This method prints the report displayed in the preview window to the printer. If you are customizing preview window controls, use this method to enable the user to preview the report in the preview window, and then, if everything looks satisfactory, to print the report to the printer (in response to a user event - button click, menu command, etc.).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEPrintWindow, Page 265

CRPEJob::SelectPrinter

Syntax

```
BOOL SelectPrinter (const char *driverName, const char *printerName,  
const char *portName, const DEVMODE *mode = 0);
```

Parameters

Parameter	Description
driverName	Specifies the name of the printer driver for the printer being selected.
printerName	Specifies the name of the printer being selected (as indicated in the Printers Control Panel).
portName	Specifies the name of the port the printer is attached to. For example: "LPT1:".
mode	A pointer to a DEVMODE structure. The default implementation of CRPEJob::SelectPrinter ignores this parameter. For more information, see <i>DEVMODE, Page 353</i> .

Remarks

This method specifies the printer and/or print characteristics for the print job. You can use this method to enable the user to select a printer other than the default printer at print time. One way of doing this is to have your application call the Windows common Print Setup dialog box.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESelectPrinter, Page 267

CRPEJob::SetDialogParentWindow

Syntax

```
BOOL SetDialogParentWindow (CWnd *parentWindow);
```

Parameters

Parameter	Description
parentWindow	A pointer to the CWnd object that is to be the parent of the preview window when the report is printed to a preview window.

Remarks

The SetDialogParentWindow function specifies the handle for the parent window for a preview window that is an MDI child.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetDialogParentWindow, Page 274

CRPEJob::SetFont

Syntax

```
BOOL SetFont (short sectionCode, short scopeCode, const char *faceName,  
short fontFamily, short fontPitch, short charSet, short pointSize,  
short isItalic, short isUnderlined, short isStruckOut, short weight);
```

Parameters

Parameter	Description
sectionCode	Specifies the section of the report for which you want to set the font. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Sets the line height for all sections.
PEP_HEADERSECTION	Sets the line height for the Page Header section.
PEP_GROUPHEADER	Sets the line height for the Group Header section.
PEP_DETAILSECTION	Sets the line height for the Details section.
PEP_GROUPFOOTER	Sets the line height for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the line height for the Grand Total section.
PEP_FOOTERSECTION	Sets the line height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
scopeCode	Specifies whether the font selected is to apply to fields, to text, or to both. To specify both, use the bitwise Or operator (). The following values are possible:

<i>Value</i>	<i>Meaning</i>
PEP_FIELDS	Sets the default font for fields in the report section specified.
PEP_TEXT	Sets the default font for all text (that has not been entered as a text field value) in the report section specified.

<i>Parameter</i>	<i>Description</i>
faceName	Specifies the actual face name of the font you want to use. The face name you pass can typically come from a font dialog box, be hard coded in the application, or be chosen by the application from the fonts supported on the printer. For example: "Times New Roman".
fontFamily	Specifies the font family for the font you want to use. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
FF_DONTCARE	No font family or family does not matter.
FF_ROMAN	Variable pitch font with serifs.
FF_SWISS	Fixed pitch font without serifs.
FF_MODERN	Fixed pitch font, with or without serifs.
FF_SCRIPT	Handwriting-like font.
FF_DECORATIVE	Fancy display font.

<i>Parameter</i>	<i>Description</i>
fontPitch	Specifies the font pitch you wish to use. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
DEFAULT_PITCH	Retains the default pitch for the font.
FIXED_PITCH	Fixed pitch, each character is the same width.
VARIABLE_PITCH	Variable pitch, the width of each character varies.

<i>Parameter</i>	<i>Description</i>
charSet	Specifies the character set you wish to use. Use one of the following values:

<i>Value</i>
ANSI_CHARSET
SYMBOL_CHARSET
HANGEUL_CHARSET
OEM_CHARSET
DEFAULT_CHARSET
SHIFTJIS_CHARSET
CHINESEBIG5_CHARSET

<i>Parameter</i>	<i>Description</i>
pointSize	Specifies the desired point size for the selected font. Use 0 to indicate no change.
isItalic	Specifies whether the font selected should be italicized. Use 1 for italics, 0 for no italics, or PEP_UNCHANGED to leave the italics as set up in the report.
isUnderlined	Specifies whether the font should be underlined. Use 1 to underline, 0 for no underline, or PEP_UNCHANGED to leave underline settings as specified in the report.
isStruckOut	Specifies whether or not the font should appear struck-out. Use 1 for strike-out, 0 for no strike out, or PEP_UNCHANGE to leave strike-out settings as specified in the report.
weight	Specifies the weight of the font. Possible values are:

<i>Value</i>
FW_DONT CARE
FW_EXTRALIGHT
FW_NORMAL
FW_SEMIBOLD
FW_EXTRABOLD
FW_ULTRALIGHT
FW_DEMIBOLD
FW_BLACK
FW_THIN
FW_LIGHT
FW_MEDIUM
FW_BOLD
FW_HEAVY
FW_REGULAR
FW_ULTRABOLD

Remarks

This method sets the font and font characteristics for the specified section. Use any time you need to change a default font at runtime in response to user input, or to specify a built-in printer font.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetFont, Page 282

CRPEJob::SetFormula

Syntax

```
BOOL SetFormula(const char *formulaName, const char *formulaText);
```

Parameters

Parameter	Description
formulaName	The name of the formula for which you want to assign new formula text.
formulaText	The actual formula text that replaces the existing formula string.

Remarks

This method sets the formula text for the specified formula. CRPEJob::SetFormula is often used with *CRPEJob::GetFormula, Page 858*, to identify and then change an existing formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if the named formula does not exist in the report, or there is an error in the formula.

Related Topics

PESetFormula, Page 287

CRPEJob::SetGraphData

Syntax

```
BOOL SetGraphData (short sectionCode, short graphN, CRPEGraphDataInfo  
*graphDataInfo);
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to change graph data information for. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																
graphN	Specifies which graph within the section you want to change graph data information for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.																
graphDataInfo	A pointer to a CRPEGraphDataInfo structure.																

Remarks

Use this method to change what data is used from your report to create a specified graph. This information includes the groups used to create the rows and columns of the graph and the summary field used to set the values of the risers in the graph.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphData, Page 289

CRPEJob::SetGraphOptions

Syntax

```
BOOL SetGraphOptions (short sectionCode, short graphN, CRPEGraphOptions  
*graphOptions);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to change display options for. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to change display options for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphOptions	A pointer to a CRPEGraphOptions structure.

Remarks

Use this function to change any of several graph options. These options include the minimum and maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphOptions, Page 291

CRPEJob::SetGraphText

Syntax

```
BOOL SetGraphText (short sectionCode, short graphN, CRPEGraphTextInfo  
*graphTextInfo);
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to change the text of. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to change the text of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphTextInfo	A pointer to a CRPEGraphTextInfo structure.

Remarks

This method allows you to set or change the text that appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphText, Page 292

CRPEJob::SetGraphType

Syntax

```
BOOL SetGraphType (short sectionCode, short graphN, short graphType);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to change the type of. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.

Value	Meaning
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

Parameter	Description
graphN	Specifies which graph within the section you want to change the type of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphType	Specifies the style of the graph you want to set. Possible values are:

Value	Meaning
PEP_SIDE_BY_SIDE_BAR_GRAPH	Side By Side bar graph
PEP_STACKED_BAR_GRAPH	Stacked bar graph
PEP_PERCENT_BAR_GRAPH	Percent bar graph
PEP_FADED_3D_SIDE_BY_SIDE_BAR_GRAPH	3D Side By Side bar graph
PEP_FADED_3D_STACKED_BAR_GRAPH	3D Stacked bar graph
PEP_FADED_3D_PERCENT_BAR_GRAPH	3D Percent bar graph
PEP_PIE_GRAPH	Pie graph
PEP_MULTIPLE_PIE_GRAPH	Multiple Pie graph
PEP_PROPORIONAL_MULTI_PIE_GRAPH	Weighted Pie graph
PEP_LINE_GRAPH	Line graph
PEP_AREA_GRAPH	Area graph
PEP_THREED_BAR_GRAPH	3D bar graph
PEP_USER_DEFINED_GRAPH	User Defined graph type

<i>Value</i>	<i>Meaning</i>
PEP_UNKNOWN_TYPE_GRAPH	Unknown graph type

Remarks

Changes the type of graph displayed for the specified graph in the specified section based on one of the available graph/chart types. Use this method to change the type of graph that is displayed in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphType, Page 294

CRPEJob::SetGroupCondition

Syntax

```
BOOL SetGroupCondition(short sectionCode, const char *conditionField,
short condition, short sortDirection);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the group section for which you want to set the group condition. Select a section from the table below:

<i>Value</i>	<i>Meaning</i>
PEP_GROUPHEADER	Sets the group condition for the Group Header section.
PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.

<i>Parameter</i>	<i>Description</i>
conditionField	Specifies the field that triggers a summary whenever its value changes. Use the name of the field as indicated in the report file.

Parameter	Description	
condition	Specifies the condition that needs to be met for Date and Boolean fields. For all field types except Date and Boolean, use PEP_GC_ANYCHANGE as the condition parameter. Select from the following:	
<i>Value (Date fields only)</i>	<i>Meaning</i>	
PEP_GC_DAILY	Triggers a grouping every time the date changes.	
PEP_GC_WEEKLY	Triggers a grouping every time the date changes from one week to the next (a week runs from Sunday through Saturday).	
PEP_GC_BIWEEKLY	Triggers a grouping every time the date changes from one two-week period to the next.	
PEP_GC_SEMIMONTHLY	Triggers a grouping every time the date changes from one half-month period to the next.	
PEP_GC_MONTHLY	Triggers a grouping every time the date changes from one month to the next.	
PEP_GC_QUARTERLY	Triggers a grouping every time the date changes from one calendar quarter to the next.	
PEP_GC_SEMIANNUALLY	Triggers a grouping every time the date changes from one half-year period to the next.	
PEP_GC_ANNUALLY	Triggers a grouping every time the date changes from one year to the next.	
<i>Value (Boolean fields only)</i>	<i>Meaning</i>	
PEP_GC_TOYES	Triggers a grouping every time the sort and group by field changes from No to Yes.	

<i>Value (Boolean fields only)</i>	<i>Meaning</i>
PEP_GC_TONO	Triggers a grouping every time the sort and group by field changes from Yes to No.
PEP_GC_EVERYYES	Triggers a grouping every time the group and sort by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the group and sort by field value is No.
PEP_GC_NEXTISYES	Triggers a grouping every time the next value in the sort and group by field is Yes.
PEP_GC_NEXTISNO	Triggers a grouping every time the next value in the sort and group by field is No.

<i>Parameter</i>	<i>Description</i>
sortDirection	Specifies one of the following sort directions:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts data in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts data in descending order (Z to A, 9 to 1).

Remarks

This method sets the condition of the grouping for the specified group section. This method can only replace the group condition for an existing group. It can not create a new group. Use this function whenever you want to change the grouping at print time, for example, to print one report grouped in several different ways.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphOptions, Page 291

CRPEJob::SetGroupSelectionFormula

Syntax

```
BOOL SetGroupSelectionFormula(const char *formulaText);
```

Parameters

Parameter	Description
formulaText	Specifies the new group selection formula to be assigned to the report.

Remarks

This method sets the formula text for the group selection formula.

CRPEJob::SetGroupSelectionFormula is often used with *CRPEJob::GetGroupSelectionFormula*, *Page 868*, to identify and then change an existing group selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong, or there is an error in the formula.

Related Topics

PESetGroupSelectionFormula, *Page 303*

CRPEJob::SetMargins

Syntax

```
BOOL SetMargins (short left, short right, short top, short bottom);
```

Parameters

Parameter	Description
left	Specifies the left margin in twips.
right	Specifies the right margin in twips.
top	Specifies the top margin in twips.

<i>Parameter</i>	<i>Description</i>
bottom	Specifies the bottom margin in twips.

NOTE: For any of the above parameters, **PM_SM_DEFAULT** can be used to specify that the report use default printer margins.

Remarks

This method sets the page margins for the print job. Use this method any time you want to allow the user to change margins.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetMargins, Page 305

CRPEJob::SetMinimumSectionHeight

Syntax

```
BOOL SetMinimumSectionHeight (short sectionCode, short height);
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section for which you want to specify the minimum height. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Sets the minimum section height for all sections.
PEP_HEADERSECTION	Sets the minimum section height for the Page Header section.
PEP_GROUPHEADER	Sets the minimum section height for the Group Header section.

<i>Value</i>	<i>Meaning</i>
PEP_DETAILSECTION	Sets the minimum section height for the Details section.
PEP_GROUPFOOTER	Sets the minimum section height for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the minimum section height for the Grand Total section.
PEP_FOOTERSECTION	Sets the minimum section height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
height	Specifies the minimum height in twips of the specified section.

Remarks

This method sets the minimum height for the specified section. For example, use this command whenever you want to specify a minimum section height for printing on pre-printed forms or printing on to any other kind of document with a fixed format.

Return Value

TRUE if the call is successful, FALSE if you have specified an invalid section or something goes wrong.

Related Topics

PESetMinimumSectionHeight, Page 307

CRPEJob::SetNDetailCopies

Syntax

```
BOOL SetNDetailCopies (short nCopies);
```

Parameters

Parameter	Description
nCopies	Specifies the number of copies of the detail section of the report to be printed.

Remarks

This method sets the number of times the Details section of the report is to be printed. For example, you can use this method to print multiple copies of labels for a customer, multiple copies of a purchase order, or multiple copies of anything set up in the Details section of the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNDetailCopies, Page 309

CRPEJob::SetNthGroupSortField

Syntax

```
BOOL SetNthGroupSortField(short sortFieldN, const char *field, short direction);
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the group sort field you want to set. The first group sort field added to the report is field 0, the second is 1, etc. If the report has N group sort fields, the function can be called with this parameter between 0 and N-1 to replace an existing group sort field. Call the function with this parameter equal to N to add a new group sort field.
field	Specifies the name of the group field to be sorted.
direction	Specifies the sort direction. The following values are possible:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts in descending order (Z to A, 9 to 1).

Remarks

This method sorts the specified group summary field in the specified direction (ascending or descending). This method does not create a new group, but will sort an existing group summary field.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthGroupSortField, Page 311

CRPEJob::SetNthParam

Syntax

```
BOOL SetNthParam (short paramN, const char *paramValue);
```

Parameters

<i>Parameter</i>	<i>Description</i>
paramN	Specifies which parameter in the stored procedure you want to set the value of. The first parameter of a stored procedure is 0, the second is 1, etc.
paramValue	Specifies the new value of the indicated parameter. This value must be a string. Please see Remarks below.

Remarks

Sets the value of a parameter in a stored procedure. Use this method when working with stored procedures in SQL database tables to set the value of a parameter in a stored procedure. When passing parameter values, all parameter values must be passed as string values. If you wish to pass a numeric value, pass the value in quotes like this: "100".

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthParam, Page 313

CRPEJob::SetNthParameterField

Syntax

```
BOOL SetNthParameterField (short parameterN, const  
CRPEParameterFieldInfo *parameterInfo);
```

Parameters

Parameter	Description
parameterN	Specifies the number of the parameter field about which you want to retrieve information.
parameterInfo	A pointer to a <i>CRPEParameterFieldInfo::CRPEParameterFieldInfo, Page 945</i> , structure.

Remarks

The SetNthParameterField function sets a value for the specified parameter field.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthParameterField, Page 315

CRPEJob::SetNthSortField

Syntax

```
BOOL SetNthSortField(short sortFieldN, const char *field, short direction);
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the sort field you want to set. The first sort field added to the report is field 0, the second is 1, etc. If the report has N sort fields, the function can be called with this parameter between 0 and N-1 to replace an existing sort field. Call the function with this parameter equal to N to add a new sort field.
field	Specifies the name of the field to be sorted.
direction	Specifies the sort direction. The following values are possible:

Value	Meaning
PEP_SF_ASCENDING	Sorts in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts in descending order (Z to A, 9 to 1).

Remarks

This method sorts report data according to the specified field and direction.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthSortField, Page 316

CRPEJob::SetNthTableLocation

Syntax

```
BOOL SetNthTableLocation(short tableN, const CRPETableLocation  
*tableLocation);
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table for which you want to set a new location.
tableLocation	A pointer to a CRPETableLocation structure.

Remarks

This method sets table location information for the specified table in the report. This method is typically combined with the *CRPEJob::GetNthTableLocation*, Page 883, method to identify the location of a table and then to change it.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PESetNthTableLocation, Page 319

CRPEJob::SetNthTableLogOnInfo

Syntax

```
BOOL SetNthTableLogOnInfo(short tableN, const CRPELogOnInfo *logonInfo,  
BOOL propagate);
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table for which you want to set log on information.

<i>Parameter</i>	<i>Description</i>
logonInfo	A pointer to a CRPELogOnInfo structure.
propagate	TRUE or FALSE value indicating whether the log on information should be used for opening all tables being used in the report.

Remarks

This method sets SQL connection information for the specified table. The propagate flag may be used to cause the change to affect all tables with similar server and database properties.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PESetNthTableLocation, Page 319

CRPEJob::SetNthTableSessionInfo

Syntax

```
BOOL SetNthTableSessionInfo(short tableN, const CRPESessionInfo * sessionInfo, BOOL propagate);
```

Parameters

<i>Parameter</i>	<i>Description</i>
tableN	0 indexed table number indicating which MS Access table in the report the session is being opened for.
sessionInfo	A pointer to a CRPESessionInfo structure.
propagate	TRUE or FALSE value indicating whether the session information should be used for opening all tables being used in the report.

Remarks

This methods sets session information for the specified Microsoft Access table. Many MS Access database tables require that a session be opened before the table can be used. Use this method to open the session.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PGetNthTableSessionInfo, Page 206

CRPEJob::SetPrintDate

Syntax

```
BOOL SetPrintDate (short year, short month, short day);
```

Parameters

Parameter	Description
year	Specifies the year of the new print date.
month	Specifies the month of the new print date.
day	Specifies the day of the new print date.

Remarks

This method sets the print date for the report. This method does not schedule the report to print at a different time or day, but only changes the date that appears in any Print Date Field that appears on the report. Use this method, for example, to post date a report when it is printed before the day it is distributed.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetPrintDate, Page 324

CRPEJob::SetPrintOptions

Syntax

```
BOOL SetPrintOptions (const CRPEPrintOptions *options);
```

Parameters

Parameter	Description
options	A pointer to a CRPEPrintOptions structure.

Remarks

This method may be used to set the print characteristics of a print job. Use this method any time you want to set the starting page number, the ending page number, the number of report copies, and/or collation instructions for a print job at runtime in response to user specifications.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetPrintOptions, Page 326

CRPEJob::SetReportTitle

Syntax

```
BOOL SetReportTitle (const char *title);
```

Parameters

Parameter	Description
title	Specifies a new title for the report.

Remarks

This method sets the report title for the print job. Use this method whenever you need to change the title of a report at print time.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetReportTitle, Page 329

CRPEJob::SetSectionFormat

Syntax

```
BOOL SetSectionFormat (short sectionCode, CRPESectionOptions *options);
```

Parameters

Parameter	Description
sectionCode	Specifies the code for the report section that you need to change format information for. Possible values are:

Value	Meaning
PEP_ALLSECTIONS	Sets the format information for all sections.
PEP_HEADERSECTION	Sets the format information for the Page Header section.
PEP_GROUPHEADER	Sets the format information for the Group Header section.
PEP_DETAILSECTION	Sets the format information for the Details section.
PEP_GROUPFOOTER	Sets the format information for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the format information for the Grand Total section.
PEP_FOOTERSECTION	Sets the format information for the Page Footer section.

Parameter	Description
options	A pointer to a CRPESectionOptions structure.

Remarks

Sets the section format settings for selected sections in the specified report to the values in the CRPESectionOptions structure. This method can be used to provide specialized formatting for printing invoices, form letters, printing to pre-printed forms, etc. It allows you to hide a section, insert a page break either before or after a section begins, reset the page number to 1 after a group value prints, prevent page breaks from spreading data from a single record over two pages, and print group values only at the bottom of a page. For a complete discussion of each of these options, see the *CRPESectionOptions::CRPESectionOptions, Page 951*, structure.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetSectionFormat, Page 331

CRPEJob::SetSelectionFormula

Syntax

```
BOOL SetSelectionFormula(const char *formulaText);
```

Parameters

Parameter	Description
formulaText	Specifies the new selection formula to be assigned to the report.

Remarks

This method sets the formula text for the record selection formula. CRPEJob::SetSelectionFormula is often used with *CRPEJob::GetSelectionFormula, Page 890*, to identify and then change an existing selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong, or there is an error in the formula.

Related Topics

PESetSelectionFormula, Page 336

CRPEJob::SetSQLQuery

Syntax

```
BOOL SetSQLQuery (const char *query);
```

Parameters

Parameter	Description
query	The text of the new SQL query to be sent to the server.

Remarks

This method sets the SQL query that will be sent to the database server. This method can be used with *CRPEJob::GetSQLQuery*, *Page 891*, to retrieve and then change the SQL query for the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetSQLQuery, *Page 337*

CRPEJob::ShowFirstPage

Syntax

```
BOOL ShowFirstPage();
```

Remarks

This method shows the first page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::ShowLastPage

Syntax

```
BOOL ShowLastPage();
```

Remarks

This method shows the last page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::ShowNextPage

Syntax

```
BOOL ShowNextPage();
```

Remarks

This method shows the next page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::ShowNthPage

Syntax

```
BOOL GetNthParamInfo (short pageN);
```

Parameters

Parameter	Description
pageN	Specifies the page number of the report that should be displayed in the preview window.

Remarks

The ShowNthPage function displays the specified page in the preview window. The report must already be generated and displayed in the preview window. This function simply changes to another page in the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShowNthPage, Page 345

CRPEJob::ShowPreviousPage

Syntax

```
BOOL ShowPreviousPage();
```

Remarks

This method shows the previous page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::ShowPrintControls

Syntax

```
BOOL ShowPrintControls(BOOL showControls);
```

Parameters

Parameter	Description
<i>showControls</i>	TRUE indicates default Print controls are to be displayed in the preview window. FALSE indicates no controls are to be displayed.

Remarks

This method toggles the display of the preview window control buttons. Use this to display default controls, or to hide default controls and provide customized controls.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShowPrintControls, Page 346

CRPEJob::Start

Syntax

```
BOOL Start();
```

Remarks

This method is used to start the processing of the print job and display the final output of that processing. This is the method that actually prints or exports the report.

Return Value

TRUE if the job is started successfully, FALSE if something goes wrong.

Related Topics

PESStartPrintJob, Page 348

CRPEJob::TestNthTableConnectivity

Syntax

```
BOOL TestNthTableConnectivity(short tableN);
```

Parameters

Parameter	Description
tableN	Specifies the number of the table for which you want to test the connection settings. The first table added to a report is numbered 0, the second is 1, etc.

Remarks

This method tests to see if a valid connection exists to the database for the specified table in the report. This method is typically used if you plan to print at a later time, but you want to test now to make sure everything is in order for logging on to the database table.

Return Value

TRUE if the database session, log on, and location information is all correct. FALSE if something is wrong.

Related Topics

PETestNthTableConnectivity, Page 349

CRPEJob::ZoomPreviewWindow

Syntax

```
BOOL ZoomPreviewWindow(short level);
```

Parameters

Parameter	Description
level	The magnification level to which the preview window is to be zoomed. Use one of the following values:
Value	Meaning
PEP_ZOOM_FULL_SIZE	Full page.
PEP_ZOOM_SIZE_FIT_ONE_SIDE	Fit one side (highest magnification).
PEP_ZOOM_SIZE_FIT_BOTH_SIDES	Fit both sides.

Remarks

This method “zooms” the preview window to the specified magnification level. Use this function when the report has been printed to a preview window and you need to set the magnification of the preview window to a specific level.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEZoomPreviewWindow, Page 351

CRPEJobInfo::CRPEJobInfo

Syntax

CRPEJobInfo();

Remarks

Constructs a CRPEJobInfo structure object. Initializes all required members of the structure.

CLASS LIBRARY SUPPORT STRUCTURES

structure CRPEExportOptions

This structure is used to get and set the export options of a print job. It is used by member function *CRPEJob::ExportTo*, *Page 856*.

Data Members

<i>Member</i>	<i>Description</i>
m_formatDLLName	Specifies the name of the format DLL that contains the export format to be used.
m_formatType	Specifies the export format to be used.
m_formatOptions	Provides additional export information specific to the format type being used.
m_destinationDLLName	Specifies the name of the destination DLL that contains the destination type to be used.
m_destinationType	Specifies the destination type of the exported report.
m_destinationOptions	Provides additional information specific to the export destination type.
m_nFormatOptionsBytes	Automatically assigned by CRPEjob::GetExportOptions. Unused by CRPEJob::SetExportOptions.
m_nDestinationOptionsBytes	Automatically assigned by CRPEjob::GetExportOptions. Unused by CRPEJob::SetExportOptions.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEExportOptions	Constructs a CRPEExportOptions structure type.

Related Topics

PEExportOptions, *Page 399*

CRPEExportOptions::CRPEExportOptions

Syntax

```
CRPEExportOptions ();
```

For example:

```
CRPEExportOptions (const char *formatDLLName, DWORD formatType, void
*formatOptions, const char *destinationDLLName, DWORD destinationType,
void *destinationOptions);
```

Parameters

Parameter	Description	
formatDLLName	Specifies the name of the format DLL that contains the export format to be used. Assigns this value to the CRPEExportOptions::m_formatDLLName member. Select a DLL from the following table:	
	<i>To export a report in this format:</i>	<i>Use this DLL:</i>
	Seagate Crystal Reports Format	uxfcr.dll
	Data Interchange Format	uxfdf.dll
	Word for Windows Format	uxfwordw.dll
	Word for DOS Format	uxfdos.dll
	WordPerfect Format	uxfdoc.dll
	Quattro Pro 5.0 (WB1) Format	uxfqp.dll
	Record Style Format (columns)	uxfrec.dll
	Rich Text Format	uxfrtf.dll
	Comma Separated Values (CSV)	uxfsepv.dll
	Tab Separated Values	uxfsepv.dll
	Character Separated Values	uxfsepv.dll
	Text Format (ASCII)	uxftext.dll
	Tab Separated Text Format	uxftext.dll
	Lotus 1-2-3 (WKS)	uxfwks.dll

Lotus 1-2-3 (WK1)	uxfwks.dll
Lotus 1-2-3 (WK3)	uxfwks.dll
Excel 2.1	uxfxls.dll
Excel 3.0	uxfxls.dll
Excel 4.0	uxfxls.dll

<i>Parameter</i>	<i>Description</i>
formatType	Specifies the export format to be used. Assigns this value to the CRPEExportOptions::m_formatType member. Select from the following values:

<i>To export a report in this format:</i>	<i>Use this for formatType:</i>
Seagate Crystal Reports Format	UXFCrystalReportType
Data Interchange Format	UXFDIFType
Word for Windows Format	UXFWordWinType
Word for DOS Format	UXFWordDosType
WordPerfect Format	UXFWordPerfectType
Quattro Pro 5.0 (WB1) Format	UXFQP5Type
Record Style Format (columns)	UXFRecordType
Rich Text Format	UXFRichTextFormatType
Comma Separated Values (CSV)	UXFCommaSeparatedType
Tab Separated Values	UXFTabSeparatedType
Character Separated Values	UXFCharSeparatedType
Text Format (ASCII)	UXFTextType
Tab Separated Text Format	UXFTabbedTextType
Lotus 1-2-3 (WKS)	UXFLotusWksType
Lotus 1-2-3 (WK1)	UXFLotusWk1Type
Lotus 1-2-3 (WK3)	UXFLotusWk3Type
Excel 2.1	UXFXls2Type
Excel 3.0	UXFXls3Type
Excel 4.0	UXFXls4Type

<i>Parameter</i>	<i>Description</i>
formatOptions	Provides additional export information specific to the format type being used. Assigns this value to the CRPEExportOptions::m_formatOptions member. This parameter is required for only certain format types. If you assign NULL to this parameter, the Crystal Report Engine will automatically prompt the user for format information if needed. Otherwise, use a format options structure from the following table:

<i>To export a report in this format:</i>	<i>Use this for formatType:</i>
Data Interchange Format	UXFDIFOptions
Record Style Format (columns)	UXFRecordStyleOptions
Comma Separated Values (CSV)	UXFCommaTabSeparated-Options
Tab Separated Values	UXFCommaTabSeparated-Options
Character Separated Values	UXFCharSeparatedOptions

<i>Parameter</i>	<i>Description</i>
destinationDLLName	Specifies the name of the destination DLL that contains the destination type to be used. Assigns this value to the CRPEExportOptions::m_destinationDLLName member. The following options are available:

<i>To export a report to this destination:</i>	<i>Use this DLL name:</i>
Disk File	uxddisk.dll
E-mail (MAPI)	uxdmapi.dll
E-mail (VIM)	uxdvim.dll

<i>Parameter</i>	<i>Description</i>
destinationType	Specifies the destination type of the exported report. Assigns this value to the CRPEExportOptions::m_destinationType member. The following values are available:

<i>To export a report to this destination:</i>	<i>Use this destination type:</i>
Disk File	UXDDDiskType

<i>To export a report to this destination:</i>	<i>Use this destination type:</i>
E-mail (MAPI)	UXDMapiType
E-mail (VIM)	UXDVIMType

<i>Parameter</i>	<i>Description</i>
destinationOptions	Provides additional information specific to the export destination type. Assigns this value to the CRPEExportOptions::m_destinationOptions member. If you assign NULL to this parameter, the Crystal Report Engine will automatically prompt the user for destination information when needed. Otherwise, use a destination options structure from the following table:

<i>To export a report to this destination:</i>	<i>Use this structure:</i>
Disk File	UXDDiskOptions
E-mail (MAPI)	UXDMAPIOptions
E-mail (VIM)	UXDVIMOptions

Remarks

Constructs a CRPEExportOptions structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEGraphDataInfo

The CRPEGraphDataInfo structure contains information on what report data is used by a graph in the report to create the values in the graph. The structure is used by *CRPEJob::GetGraphData*, *Page 858*, to retrieve information regarding an existing graph and by *CRPEJob::SetGraphData*, *Page 903*, to change the data used by an existing graph.

Data Members

<i>Member</i>	<i>Description</i>
m_rowGroupN	Specifies which group number in the report is used to create the values in the rows of the graph.
m_colGroupN	Specifies which group number in the report is used to create the values in the columns of the graph.

<i>Member</i>	<i>Description</i>
m_summarizedFieldN	Specifies which summary field in the report is used to set the values of the risers in the graph. Summary fields are numbered in order of their creation.
m_graphDirection	Specifies the graphing direction for cross-tab reports. For normal group/total report, the direction, is always PE_GRAPH_COLS_ONLY.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEGraphDataInfo	Constructs a CRPEGraphDataInfo structure type.

Related Topics

PEGraphDataInfo, Page 410

CRPEGraphDataInfo::CRPEGraphDataInfo

Syntax

```
CRPEGraphDataInfo ();
```

For example:

```
CRPEGraphDataInfo (short rowGroupN, short colGroupN, short
summarizedFieldN, short graphDirection);
```

Parameters

<i>Parameter</i>	<i>Description</i>
rowGroupN	Specifies which group number in the report is used to create the values in the rows of the graph. Assigns this value to the CRPEGraphDataInfo::m_rowGroupN member variable.
colGroupN	Specifies which group number in the report is used to create the values in the columns of the graph. Assigns this value to the CRPEGraphDataInfo::m_colGroupN member variable.
summarizedFieldN	Specifies which summary field in the report is used to set the values of the risers in the graph. Summary fields are numbered in order of their creation. Assigns this value to the CRPEGraphDataInfo::m_summarizedFieldN member variable.

<i>Parameter</i>	<i>Description</i>												
graphDirection	Specifies the graphing direction for cross-tab reports. For normal group/total report, the direction, is always PEP_GRAPH_COLS_ONLY. Assigns this value to the CRPEGraphDataInfo::m_graphDirection member variable. Possible values are:												
	<table border="1"> <thead> <tr> <th><i>Value</i></th><th><i>Meaning</i></th></tr> </thead> <tbody> <tr> <td>PEP_GRAPH_ROWS_ONLY</td><td>Use only row values in graph.</td></tr> <tr> <td>PEP_GRAPH_COLS_ONLY</td><td>Use only column values in graph.</td></tr> <tr> <td>PEP_GRAPH_MIXED_ROW_COL</td><td>Graph by row values, then by column values.</td></tr> <tr> <td>PEP_GRAPH_MIXED_COL_ROW</td><td>Graph by column values, then by row values.</td></tr> <tr> <td>PEP_GRAPH_UNKNOWN_DIRECTION</td><td>The direction of the graph is unknown.</td></tr> </tbody> </table>	<i>Value</i>	<i>Meaning</i>	PEP_GRAPH_ROWS_ONLY	Use only row values in graph.	PEP_GRAPH_COLS_ONLY	Use only column values in graph.	PEP_GRAPH_MIXED_ROW_COL	Graph by row values, then by column values.	PEP_GRAPH_MIXED_COL_ROW	Graph by column values, then by row values.	PEP_GRAPH_UNKNOWN_DIRECTION	The direction of the graph is unknown.
<i>Value</i>	<i>Meaning</i>												
PEP_GRAPH_ROWS_ONLY	Use only row values in graph.												
PEP_GRAPH_COLS_ONLY	Use only column values in graph.												
PEP_GRAPH_MIXED_ROW_COL	Graph by row values, then by column values.												
PEP_GRAPH_MIXED_COL_ROW	Graph by column values, then by row values.												
PEP_GRAPH_UNKNOWN_DIRECTION	The direction of the graph is unknown.												

Remarks

Constructs a CRPEGraphDataInfo structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEGraphOptions

The CRPEGraphOptions structure contains information on several options available with graphs and charts. This structure is used by *CRPEJob::GetGraphOptions*, Page 860, to find out what options have been set for a graph, and it is used by *CRPEJob::SetGraphOptions*, Page 905, to change the options for a graph.

Data Members

<i>Member</i>	<i>Description</i>
m_graph.MaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted.
m_graph.MinValue	Specifies the minimum value that will appear in the graph. Any graph values below this value are not charted.
m_showDataValue	Specifies whether or not to display the numeric value associated with each riser on the chart. If set to TRUE (1), a value appears in the graph for each riser.

<i>Member</i>	<i>Description</i>
m_showGridLine	Specifies whether or not to display grid lines on the graph.
m_verticalBars	Specifies whether to display the bars in a bar graph vertically or horizontally.
m_showLegend	Specifies whether or not to display the graph legend.
m_fontFaceName	Specifies the font for all text and values in the entire graph.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEGraphOptions	Constructs a CRPEGraphOptions structure type.

Related Properties

PEGraphOptions, Page 413

CRPEGraphOptions::CRPEGraphOptions

Syntax

```
CRPEGraphOptions () ;
```

For example:

```
CRPEGraphOptions (double graph.MaxValue, double graph.MinValue, BOOL
showDataValue, BOOL showGridLine, BOOL verticalBars, BOOL showLegend,
const char *fontFaceName);
```

Parameters

<i>Parameter</i>	<i>Description</i>
graph.MaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted. Assigns this value to the CRPEGraphOptions::m_graph.MaxValue member variable.
graph.MinValue	Specifies the minimum value that will appear in the graph. Any graph values below this value are not charted. Assigns this value to the CRPEGraphOptions::m_graph.MinValue member variable.

<i>Parameter</i>	<i>Description</i>
showDataValue	Specifies whether or not to display the numeric value associated with each riser on the chart. If set to TRUE, a value appears in the graph for each riser. Assigns this value to the CRPEGraphOptions::m_showDataValue member variable.
showGridLine	Specifies whether or not to display grid lines on the graph. Assigns this value to the CRPEGraphOptions::m_showGridLine member variable.
verticalBars	Specifies whether to display the bars in a bar graph vertically or horizontally. Assigns this value to the CRPEGraphOptions::m_verticalBars member variable.
showLegend	Specifies whether or not to display the graph legend. Assigns this value to the CRPEGraphOptions::m_showLegend member variable.
fontFaceName	Specifies the font for all text and values in the entire graph. Assigns this value to the CRPEGraphOptions::m_fontFaceName member variable.

Remarks

Constructs a CRPEGraphOptions structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEGraphTextInfo

The CRPEGraphTextInfo structure contains information about the text that appears with a graph. This structure is used with the methods *CRPEJob::GetGraphText*, *Page 861*, and *CRPEJob::SetGraphText*, *Page 906*.

Data Members

<i>Member</i>	<i>Description</i>
m_graphTitle	Specifies the main title text that will appear above your graph.
m_graphSubTitle	Specifies the subtitle text that will appear directly under the main title.
m_graphFootNote	Specifies the footnote text that will appear under your graph.
m_graphGroupsTitle	Specifies the title of the groups which are being graphed.
m_graphSeriesTitle	Specifies the title of the series which is being graphed.
m_graphXAxisTitle	Specifies the title text that will appear for the X axis. Not valid for Pie graphs.

<i>Member</i>	<i>Description</i>
m_graphYAxisTitle	Specifies the title text that will appear for the Y axis. Not valid for Pie graphs.
m_graphZAxisTitle	Specifies the title text that will appear for the Z axis. This value is only valid for 3D graphs.

Construction

<i>Construction</i>	<i>Description</i>
CRPEGraphTextInfo	Constructs a CRPEGraphTextInfo structure type.

Related Topics

PEGraphTextInfo, Page 415

CRPEGraphTextInfo::CRPEGraphTextInfo

Syntax

```
CRPEGraphTextInfo();
```

For example:

```
CRPEGraphTextInfo (const char *graphSubTitle, const char
*graphFootNote, const char *graphGroupsTitle, const char
*graphSeriesTitle, const char *graphXAxisTitle, const char
*graphYAxisTitle, const char *graphZAxisTitle);
```

Parameters

<i>Parameter</i>	<i>Description</i>
graphTitle	Specifies the main title text that will appear above your graph. Assigns this title to the CRPEGraphTextInfo::m_graphTitle member variable.
graphSubTitle	Specifies the subtitle text that will appear directly under the main title. Assigns this title to the CRPEGraphTextInfo::m_graphSubTitle member variable.
graphFootNote	Specifies the footnote text that will appear under your graph. Assigns this text to the CRPEGraphTextInfo::m_graphFootNote member variable.

<i>Parameter</i>	<i>Description</i>
graphGroupsTitle	Specifies the title of the groups which are being graphed. Assigns this title to the CRPEGraphTextInfo::m_graphGroupsTitle member variable.
graphSeriesTitle	Specifies the title of the series which is being graphed. Assigns this title to the CRPEGraphTextInfo::m_graphSeriesTitle member variable.
graphXAxisTitle	Specifies the title text that will appear for the X axis. Not valid for Pie graphs. Assigns this title to the CRPEGraphTextInfo::m_graphXAxisTitle member variable.
graphYAxisTitle	Specifies the title text that will appear for the Y axis. Not valid for Pie graphs. Assigns this title to the CRPEGraphTextInfo::m_graphYAxisTitle member variable.
graphZAxisTitle	Specifies the title text that will appear for the Z axis. Only valid for 3D graphs. Assigns this title to the CRPEGraphTextInfo::m_graphZAxisTitle member variable.

Remarks

Constructs a CRPEGraphTextInfo structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEJobInfo

This structure is used by *CRPEJob::GetJobStatus*, *Page 869*, member function. Status information concerning the job is returned through this structure.

Data Members

<i>Member</i>	<i>Description</i>
m_numRecordsRead	Specifies the number of records actually processed.
m_numRecordsSelected	Specifies the number of records selected for inclusion in the report out of the total number of records read.
m_numRecordsPrinted	Specifies the number of records actually printed.
m_displayPageN	Specifies the page number of the currently displayed page in the preview window.
m_latestPageN	Specifies the number of the last page that is currently available. Once the printing is complete, this value is the number of the last page.

<i>Member</i>	<i>Description</i>
m_startPageN	Specifies the number of the starting page. The value is normally 1.
m_printEnded	Specifies whether or not the printing process is completed.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEJobInfo	Constructs a CRPEJobInfo structure type.

Related Topics

PEJobInfo, Page 423

structure CRPELogOnInfo

This structure is used by member functions:

- *CRPEngine::LogOnServer, Page 845*
- *CRPEngine::LogOffServer, Page 844*
- *CRPEJob::SetNthTableLogOnInfo, Page 919*
- *CRPEJob::SetNthTableLogOnInfo, Page 919*

The structure is used to contain SQL connection information.

Data Members

<i>Member</i>	<i>Description</i>
m_serverName	Specifies the logon name for the server used to create the report.
m_databaseName	Specifies the logon name for the database used to create the report.
m(userID)	Specifies the user ID necessary to log on to the server.
m_password	Specifies the password necessary to log on to the server.

Construction

<i>Constructor</i>	<i>Description</i>
CRPELogOnInfo	Constructs a CRPELogOnInfo structure type.

Related Topics

PLogOnInfo, Page 425

CRPELogOnInfo::CRPELogOnInfo

Syntax

```
CRPELogOnInfo ();
```

For example:

```
CRPELogOnInfo (const char *serverName, const char *databaseName, const  
char *userID, const char *password);
```

Parameters

Parameter	Description
serverName	Specifies the logon name for the server used to create the report. Assigns this value to the CRPELogOnInfo::m_serverName member.
databaseName	Specifies the logon name for the database used to create the report. Assigns this value to the CRPELogOnInfo::m_databaseName member.
userID	Specifies the user ID necessary to log on to the server. Assigns this value to the CRPELogOnInfo::m(userID) member.
password	Specifies the password necessary to log on to the server. Assigns this value to the CRPELogOnInfo::m_password member.

Remarks

Constructs a CRPELogOnInfo structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEParameterFieldInfo

The CRPEParameterFieldInfo structure contains information related to parameter fields in a report. This structure is used by *CRPEJob::GetNthParameterField*, Page 880, to get information about a specific parameter field, and it is used by *CRPEJob::SetNthParameterField*, Page 917, to change a specific parameter field.

Data Members

<i>Member</i>	<i>Description</i>																
m_ValueType	Specifies the data type of the parameter field. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, string, date/time, and time. Use the appropriate constant from the list below:																
	<table border="1"> <thead> <tr> <th><i>Data Type</i></th><th><i>Constant</i></th></tr> </thead> <tbody> <tr> <td>Number</td><td>PEP_PF_NUMBER</td></tr> <tr> <td>Currency</td><td>PEP_PF_CURRENCY</td></tr> <tr> <td>Boolean</td><td>PEP_PF_BOOLEAN</td></tr> <tr> <td>Date</td><td>PEP_PF_DATE</td></tr> <tr> <td>String</td><td>PEP_PF_STRING</td></tr> <tr> <td>Date/Time</td><td>PEP_PF_DATETIME</td></tr> <tr> <td>Time</td><td>PEP_PF_TIME</td></tr> </tbody> </table>	<i>Data Type</i>	<i>Constant</i>	Number	PEP_PF_NUMBER	Currency	PEP_PF_CURRENCY	Boolean	PEP_PF_BOOLEAN	Date	PEP_PF_DATE	String	PEP_PF_STRING	Date/Time	PEP_PF_DATETIME	Time	PEP_PF_TIME
<i>Data Type</i>	<i>Constant</i>																
Number	PEP_PF_NUMBER																
Currency	PEP_PF_CURRENCY																
Boolean	PEP_PF_BOOLEAN																
Date	PEP_PF_DATE																
String	PEP_PF_STRING																
Date/Time	PEP_PF_DATETIME																
Time	PEP_PF_TIME																
<i>Member</i>	<i>Description</i>																
m_DefaultValueSet	Indicates whether or not a default value was set for the parameter field when the parameter field was created or modified in Seagate Crystal Reports. The value can be either TRUE (1) if the field was given a default value or FALSE (0) if it was not.																
m_CurrentValueSet	If <i>CRPEJob::Start</i> , <i>Page 928</i> , has not been called, this parameter will equal 0 (FALSE). Since the print job hasn't been started, there is no current value set by code or by the user. If CRPEJob::Start has been called, and the user has entered a parameter and said OK to the prompt, this parameter will equal 1 (TRUE) if the user enters a value at the prompt and 0 (FALSE) if the user clicks Cancel.																
m_Name	Specifies the name of the parameter field assigned by the user when the parameter was inserted into the report or the name that was set from code using CRPEJob::SetNthParameterField.																
m_Prompt	Specifies the prompting text, if any, that appears when the user runs the report for the first time or refreshes the data. This will be either the prompt assigned by the user when the parameter field was inserted into the report or the prompt that was set from code using CRPEJob::SetNthParameterField.																
m_DefaultValue	Specifies the default value assigned to the parameter field if one was set. If the m_DefaultValueSet member is FALSE, this value is meaningless.																

<i>Member</i>	<i>Description</i>
m_CurrentValue	If CRPEJob::Start has not been called, this value does not apply. There is no current value as yet set by code or by the user. If CRPEJob::Start has been called, this will be the value entered by the user when the user was prompted. If the user clicked Cancel, this value is meaningless.

Construction

<i>Construction</i>	<i>Description</i>
CRPEParameterFieldInfo	Constructs a CRPEParameterFieldInfo structure type.

Related Topics

PEParameterFieldInfo, Page 427

CRPEParameterFieldInfo::CRPEParameterFieldInfo

Syntax

```
CRPEParameterFieldInfo ();
```

For example:

```
CRPEParameterFieldInfo (WORD ValueType, WORD DefaultValueSet, WORD
CurrentValueSet, const _TCHAR *Name, const _TCHAR *Prompt, const _TCHAR
*DefaultValue, const _TCHAR *CurrentValue);
```

Parameters

<i>Parameter</i>	<i>Description</i>
ValueType	Specifies the data type of the parameter field. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, string, date/time, and time. Assigns this value to the CRPEParameterFieldInfo::m_ValueType member variable. Use the appropriate constant from the list below:

<i>Data Type</i>	<i>Constant</i>
Number	PEP_PF_NUMBER
Currency	PEP_PF_CURRENCY

Boolean	PEP_PF_BOOLEAN
Date	PEP_PF_DATE
String	PEP_PF_STRING
Date/Time	PEP_PF_DATETIME
Time	PEP_PF_TIME

Parameter	Description
DefaultValueSet	Specifies whether or not a default value is set for the parameter field. The value can be either TRUE (1) if you want to change the default value or FALSE (0) if you do not. Assigns this value to the CRPEParameterFieldInfo::m_DefaultValueSet member variable.
CurrentValueSet	Indicates whether or not a new value is being assigned to the parameter field. Use TRUE (1) to indicate that a new value is set, FALSE (0) to indicate no change. Assigns this value to the CRPEParameterFieldInfo::m_CurrentValueSet member variable.
Name	Specifies the name of the parameter field to be assigned a new value. Assigns this value to the CRPEParameterFieldInfo::m_Name member variable.
Prompt	Specifies the prompting text, if any, that will appear when the user runs the report for the first time or refreshes the data. Assigns this value to the CRPEParameterFieldInfo::m_Prompt member variable.
DefaultValue	Sets the default value assigned to the parameter field. If the DefaultValueSet member is FALSE, this value is meaningless. Assigns this value to the CRPEParameterFieldInfo::m_DefaultValue member variable.
CurrentValue	Assigns a value to the parameter field. If CurrentValueSet is FALSE, this value is meaningless. Assigns this value to the CRPEParameterFieldInfo::m_CurrentValue member variable.

Remarks

Constructs a CRPEParameterFieldInfo structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPEParameterInfo

The CRPEParameterInfo structure contains information about a parameter for a stored procedure in an SQL database. This structure is used by *CRPEJob::GetNthParamInfo*, *Page 880*.

Data Members

<i>Member</i>	<i>Description</i>
m_Type	Specifies the data type of the parameter. The Crystal Report Engine supports the following data types: number, currency, Boolean, date, string, date/time, and time. Use the appropriate constant from the list below:

<i>Data Type</i>	<i>Constant</i>
Number	PEP_PF_NUMBER
Currency	PEP_PF_CURRENCY
Boolean	PEP_PF_BOOLEAN
Date	PEP_PF_DATE
String	PEP_PF_STRING
Date/Time	PEP_PF_DATETIME
Time	PEP_PF_TIME

<i>Member</i>	<i>Description</i>
m_Name	Specifies the name of the parameter in a null-terminated string.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEParameterInfo	Constructs a CRPEParameterInfo structure type.

Related Topics

PEParameterInfo, *Page 432*

CRPEParameterInfo::CRPEParameterInfo

Syntax

```
CRPEParameterInfo();
```

Remarks

Constructs a CRPEParameterInfo structure object. Assigns default values to the members of the structure.

structure CRPEPrintOptions

This structure is used to set the print characteristics of a report. It is used with *CRPEJob::SetPrintOptions*, *Page 922*, member function.

Data Members

<i>Member</i>	<i>Description</i>
m_startPageN	Specifies the first page that you want to print.
m_stopPageN	Specifies the last page that you want to print.
m_nReportCopies	Specifies the number of copies that you want to print.
m_collation	Indicates whether or not you want the copies of the report to be collated (if you are printing multiple copies of a multiple page report).

Construction

<i>Constructor</i>	<i>Description</i>
CRPEPrintOptions	Constructs a CRPEPrintOptions structure type.

Related Topics

PEPrintOptions, *Page 435*

CRPEPrintOptions::CRPEPrintOptions

Syntax

```
CRPEPrintOptions ();
```

For example:

```
CRPEPrintOptions (unsigned short startPageN, unsigned short stopPageN,  
unsigned short nReportCopies, unsigned short collation);
```

Parameters

Parameter	Description
startPageN	Specifies the first page that you want to print. Assigns this value to the CRPEPrintOptions::m_startPageN member.
stopPageN	Specifies the last page that you want to print. Assigns this value to the CRPEPrintOptions::m_stopPageN member.
nReportCopies	Specifies the number of copies that you want to print. Assigns this value to the CRPEPrintOptions::m_nReportCopies member.
collation	Indicates whether or not you want the copies of the report to be collated (if you are printing multiple copies of a multiple page report). Assigns this value to the CRPEPrintOptions::m_collation member. Possible values are:

Value	Meaning
PEP_UNCOLLATED	Prints multiple copies of a multiple page report uncollated (Page order = 1, 1, 1, 2, 2, 2, 3, 3, 3, etc.)
PEP_COLLATED	Prints multiple copies of a multiple page report collated (Page order = 1, 2, 3..., 1, 2, 3..., etc.)
PEP_DEFAULTCOLLATION	Prints multiple copies of a multiple page report using the collation settings specified in the report.

Remarks

Constructs a CRPEPrintOptions structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPESectionOptions

This structure contains specifications for formatting selected report sections. This information is used by the methods *CRPEJob::GetSectionFormat*, *Page 888*, and *CRPEJob::SetSectionFormat*, *Page 923*.

Data Members

<i>Member</i>	<i>Description</i>
m_visible	Specifies whether or not the selected section is to be visible. Use TRUE to keep the section visible, FALSE to hide the section.
m_newPageBefore	Specifies whether or not the Crystal Report Engine is to insert a page break before the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break.
m_newPageAfter	Specifies whether or not the Crystal Report Engine is to insert a page break after the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break.
m_keepTogether	Specifies whether or not the Crystal Report Engine is to keep all lines of the section together, either on the current page (if there is room), or on the next page. Pass TRUE to keep the lines together, FALSE to allow the Crystal Report Engine to split section data from one page to the next if necessary.
m_suppressBlankLines	Specifies whether or not the Crystal Report Engine is to eliminate lines from your report that are blank due to fields being suppressed (zeroes, duplicates, and hidden fields). TRUE eliminates blank lines, FALSE retains them.
m_resetPageNAfter	Specifies whether or not the Crystal Report Engine is to reset the page number to one (1) for the following page, after it prints a group total. Use TRUE to reset the page number, FALSE to keep standard numbering.
m_printAtBottomOfPage	Specifies whether or not the Crystal Report Engine is to cause each group summary value to print only at the bottom of a page. Use TRUE to print summaries at the bottom of the page, FALSE to print summaries immediately after the group.
m_backgroundColor	Specifies the RGB color value contained in the <i>COLORREF</i> , Page 353 , structure.
m_underlaySection	Indicates whether or not the specified section is to underlay the following section(s).

Construction

<i>Constructor</i>	<i>Description</i>
CRPESectionOptions	Constructs a CRPESectionOptions structure type.

Related Topics

PESectionOptions, [Page 444](#)

CRPESectionOptions::CRPESectionOptions

Syntax

```
CRPESectionOptions ();
```

For example:

```
CRPESectionOptions (short visible, short newPageBefore, short  
newPageAfter, short keepTogether, short suppressBlankLines, short  
resetPageNAfter, short printAtBottomOfPage, COLORREF backgroundColor,  
short underlaySection);
```

Parameters

Parameter	Description
visible	Specifies whether or not the selected section is to be visible. Use TRUE to keep the section visible, FALSE to hide the section. Assigns this value to the CRPESectionOptions::m_visible member.
newPageBefore	Specifies whether or not the Crystal Report Engine is to insert a page break before the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break. Assigns this value to the CRPESectionOptions::m_newPageBefore member.
newPageAfter	Specifies whether or not the Crystal Report Engine is to insert a page break after the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break. Assigns this value to the CRPESectionOptions::m_newPageAfter member.
keepTogether	Specifies whether or not the Crystal Report Engine is to keep all lines of the section together, either on the current page (if there is room), or on the next page. Pass TRUE to keep the lines together, FALSE to allow the Crystal Report Engine to split section data from one page to the next if necessary. Assigns this value to the CRPESectionOptions::m_keepTogether member.
suppressBlankLines	Specifies whether or not the Crystal Report Engine is to eliminate lines from your report that are blank due to fields being suppressed (zeroes, duplicates, and hidden fields). TRUE eliminates blank lines, FALSE retains them. Assigns this value to the CRPESectionOptions::m_suppressBlankLines member.

<i>Parameter</i>	<i>Description</i>
resetPageNAfter	Specifies whether or not the Crystal Report Engine is to reset the page number to one (1) for the following page, after it prints a group total. Use TRUE to reset the page number, FALSE to keep standard numbering. Assigns this value to the CRPESectionOptions::m_resetPageNAfter member.
printAtBottomOfPage	Specifies whether or not the Crystal Report Engine is to cause each group summary value to print only at the bottom of a page. Use TRUE to print summaries at the bottom of the page, FALSE to print summaries immediately after the group. Assigns this value to the CRPESectionOptions::m_printAtBottomOfPage member.
backgroundColor	Specifies the RGB color value contained in the <i>COLORREF</i> , <i>Page 353</i> , structure. Assigns this value to the CRPESectionOptions::m_backgroundColor member.
underlaySection	Indicates whether or not the specified section is to underlay the following section(s). Assigns this value to the CRPESectionOptions::m_underlaySection member.

Remarks

- Constructs a CRPESectionOptions structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.
- Use PEP_UNCHANGED with any parameter to let the Crystal Report Engine leave the setting as it is specified in the report file.

structure CRPESessionInfo

This structure is used to get and set the session information (user ID and password) for password protected Microsoft Access databases. It is used with *CRPEJob::GetNthTableSessionInfo*, *Page 884*, and *CRPEJob::SetNthTableSessionInfo*, *Page 920*, member functions.

Data Members

<i>Member</i>	<i>Description</i>
m(userID)	Specifies the user ID needed for logging on to the MS Access system.
m(password)	Specifies the password needed for logging on to the MS Access system.
m(sessionHandle)	The handle to the current MS Access session.

Construction

<i>Constructor</i>	<i>Description</i>
CRPESessionInfo	Constructs a CRPESessionInfo structure type.

Related Topics

PESessionInfo, Page 448

CRPESessionInfo::CRPESessionInfo

Syntax

```
CRPESessionInfo ( );
```

For example:

```
CRPESessionInfo (const char *userID, const char *password, DWORD  
sessionHandle);
```

Parameters

<i>Parameters</i>	<i>Description</i>
userID	Specifies the user ID needed for logging on to the MS Access system. Assigns this value to the CRPESessionInfo::m(userID) member.
password	Specifies the password needed for logging on to the MS Access system. Assigns this value to the CRPESessionInfo::m(password) member.
sessionHandle	The handle to the current MS Access session. Assigns this value to the CRPESessionInfo::m(sessionHandle) member.

Remarks

Constructs a CRPESessionInfo structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPESubreportInfo

The CRPESubreportInfo structure contains information on a subreport that appears in the main report. This structure is used by *CRPEJob::GetSubreportInfo*, *Page 891*, to retrieve information about a subreport.

Data Members

<i>Member</i>	<i>Description</i>
m_name	Specifies the name of the subreport. This is the name assigned to the subreport when it was first created.

Construction

<i>Constructor</i>	<i>Description</i>
CRPESubreportInfo	Constructs a CRPESubreportInfo structure type.

Related Topics

PESStopEventInfo, *Page 453*

CRPESubreportInfo::CRPESubreportInfo

Syntax

```
CRPESubreportInfo();
```

Remarks

Constructs a CRPESubreportInfo structure object. Assigns default values to the members of the structure.

structure CRPETableLocation

This structure is used to get and set table location information. It is used with *CRPEJob::GetNthTableLocation*, *Page 883*, and *CRPEJob::SetNthTableLocation*, *Page 919*, functions.

Data Members

Member	Description
m_location	Specifies the database location.

Construction

Constructor	Description
CRPETableLocation	Constructs a CRPETableLocation structure type.

Related Topics

PETableLocation, Page 457

CRPETableLocation::CRPETableLocation

Syntax

```
CRPETableLocation ( );
```

For example:

```
CRPETableLocation (const char *location);
```

Parameters

Parameter	Description
location	Specifies the database location. Assigns this value to the CRPETableLocation::m_location member.

Remarks

Constructs a CRPETableLocation structure object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

structure CRPETableType

This structure is used to determine information about a specific table used in the report. It is used by *CRPEJob::GetNthTableSessionInfo, Page 884*.

Data Members

<i>Member</i>	<i>Description</i>
m_dllName	Specifies the name of the appropriate database DLL for the table of interest.
m_descriptiveName	Specifies the name of the table of interest.
m_dbType	Specifies the type of database that contains the table of interest.

Construction

<i>Constructor</i>	<i>Description</i>
CRPETableType	Constructs a CRPETableType structure type.

Related Topics

PETableType, Page 459

CRPETableType::CRPETableType

Syntax

```
CRPETableType ();
```

Remarks

Constructs a CRPETableType structure object. Initializes all members of the structure.

REPORT ENGINE CLASSES

class CRPEngine: public CObject

The CRPEngine class is designed so that there should only be one CRPEngine object in the entire application. The CRPEngine object contains methods that are common to all print jobs (i.e., SQL connections, version information, etc.). More importantly, it is responsible for creating and managing all CRPEJob objects. It is the CRPEJob object that allows you access to the attributes of a print job.

In order to open a particular report it is first necessary to have an open Crystal Report Engine object in the application. You may then call the CRPEngine::OpenPrintJob member function specifying the report file name to open. If successful, you will be returned a pointer to a CRPEJob object.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEngine	Constructs a CRPEngine object.

12

Crystal Class Library for NewEra Reference

What you will find in this chapter...

CLASS CRPEngine, Page 960

CLASS CRPEExportOptions, Page 1043

CLASS CRPEGraphDataInfo, Page 1047

CLASS CRPEGraphOptions, Page 1049

CLASS CRPEGraphTextInfo, Page 1050

CLASS CRPEJobInfo, Page 1052

CLASS CRPELogOnInfo, Page 1053

CLASS CRPEPrintOptions, Page 1055

CLASS CRPESectionOptions, Page 1056

CLASS CRPESessionInfo, Page 1059

CLASS CRPETableLocation, Page 1060

CLASS CRPETableType, Page 1061

CLASS CRPEngine

The CRPEngine class is designed so that there should only be one CRPEngine object in the entire application. The CRPEngine class contains methods that are common to all print jobs (i.e., SQL connections, version information, etc.). More importantly, it is responsible for creating and managing all CRPEJob objects. It is the CRPEJob object that allows you access to the attributes of a print job.

In order to open a particular report it is first necessary to have an open Crystal Report Engine object in the application. You may then call *CRPEngine::OpenJob*, *Page 967*, specifying the report file name to open. If successful, you will be returned a pointer to a CRPEJob object.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEngine	Constructs a CRPEngine object.

CRPEngine::CanClose

Syntax

```
CanClose () RETURNING BOOLEAN
```

Remarks

Checks to see whether or not the Crystal Report Engine is busy. Errors can occur in your application or on a system if you attempt to close the Crystal Report Engine while it is processing a print job. Use this method before attempting to close the Crystal Report Engine in an application (for example, when the user tries to exit) to determine if the Report Engine can be closed safely.

Return Value

TRUE if the Crystal Report Engine can be closed, FALSE if the Report Engine is still busy.

CRPEngine::Close

Syntax

```
Close() RETURNING VOID
```

Remarks

This method closes the Crystal Report Engine.

Related Topics

PECloseEngine, Page 127

CRPEngine::CRPEngine

Syntax

```
CRPEngine ( v_open BOOLEAN : FALSE );
```

Parameters

Parameter	Description
v_open	Indicates whether or not the Crystal Report Engine should be opened when the CRPEngine object is created.

Remarks

This is the constructor for the class. If open is true, the actual Seagate Crystal Reports DLL is opened (crpe32.dll). Print jobs may only be opened if the engine itself is open. If originally opened with open = FALSE, the engine may be opened later with *CRPEngine::Open, Page 967*.

CRPEngine::GetEngineStatus

Syntax

```
GetEngineStatus() RETURNING SMALLINT
```

Remarks

This method may be used to get the current engine status. Possible values returned are:

Value	Description
engineOpen	The Crystal Report Engine is currently open.
engineClosed	The Crystal Report Engine has been closed, or is not yet open.
engineMissing	No Crystal Report Engine is available. Make sure the Crystal Report Engine DLL is located in a directory that appears in your PATH.

Return Value

A value of the Status enumerated type. Indicates the current status of the Crystal Report Engine.

CRPEngine::GetErrorCode

Syntax

```
GetErrorCode () RETURNING SMALLINT
```

Remarks

This method returns the current error code of the Crystal Report Engine. When a call to another function fails, this call gets the error code that was generated so you can take some action based on that error code.

Return Value

The current Crystal Report Engine error code. 0 if no error has occurred.

Related Topics

PEGetErrorCode, Page 152

CRPEngine::GetErrorText

Syntax

```
GetErrorText () RETURNING ixString
```

Remarks

This method returns a descriptive Crystal Report Engine error message.

Return Value

A CString object containing the text for the current error.

Related Topics

PEGetErrorText, Page 153

CRPEngine::GetNPrintJobs

Syntax

```
GetNPrintJobs () RETURNING SMALLINT;
```

Remarks

This method returns the number of print job (CRPEJob) objects that currently exist for the current CRPEngine object.

Return Value

The number of CRPEJob objects currently open. Returns -1 if an error occurs.

CRPEngine::GetVersion

Syntax

```
GetVersion (versionRequested SMALLINT) RETURNING SMALLINT
```

Parameters

<i>Parameter</i>	<i>Description</i>						
<i>versionRequested</i>	Specifies whether the version number of the Crystal Report Engine or the Report Engine DLL is being requested. Possible values are: <table border="1"><thead><tr><th><i>Value</i></th><th><i>Meaning</i></th></tr></thead><tbody><tr><td>PEP_GV_DLL</td><td>Returns the version of the DLL.</td></tr><tr><td>PEP_GV_ENGINE</td><td>Returns the version of the Crystal Report Engine.</td></tr></tbody></table>	<i>Value</i>	<i>Meaning</i>	PEP_GV_DLL	Returns the version of the DLL.	PEP_GV_ENGINE	Returns the version of the Crystal Report Engine.
<i>Value</i>	<i>Meaning</i>						
PEP_GV_DLL	Returns the version of the DLL.						
PEP_GV_ENGINE	Returns the version of the Crystal Report Engine.						

Remarks

This method returns the version number of the Crystal Report Engine DLL (CRPE32.DLL) or the version of the Report Engine itself. The high-order byte is the major version number and the low-order byte is the minor version number.

This method can be used whenever you build functionality into a report that may not be available in

earlier versions of the Crystal Report Engine and you need to verify the version number first. The function can be a handy safeguard for users who have more than one version of the Crystal Report Engine with the older version earlier in the path than the new version.

Return Value

The version number of the currently used Crystal Report Engine or Report Engine DLL.

Related Topics

PEGetVersion, Page 231

CRPEngine::LogOffServer

Syntax

```
LogOffServer(dllName CHAR (*), const CRPELogOnInfo logOnInfo) RETURNING  
BOOLEAN
```

Parameters

Parameter	Description
<i>dllName</i>	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to.
<i>logOnInfo</i>	Specifies a pointer to a CRPELogOnInfo class.

Remarks

This method closes an existing connection to a SQL server. Connection information is provided through the CRPELogOnInfo class.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PLogOffServer, Page 240

CRPEngine::LogOnServer

Syntax

```
LogOnServer (dllName CHAR(*), logOnInfo CRPELogOnInfo) RETURNING  
BOOLEAN;
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table you want to log on to.
logOnInfo	Specifies a pointer to a CRPELogOnInfo class.

Remarks

This method opens a connection to a SQL server. More than one print job may use this connection to a SQL server. Information required to establish a connection is provided through the *CRPELogOnInfo::CRPELogOnInfo*, *Page 1054*, class.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PLogOnServer, *Page 241*

CRPEngine::LogOnSQLServerWithPrivateInfo

Syntax

```
LogOnSQLServerWithPrivateInfo (dllName CHAR (*), FOREIGN privateInfo)  
RETURNING BOOLEAN
```

Parameters

Parameter	Description
dllName	Specifies the name of the Seagate Crystal Reports DLL that was used when establishing a connection to the Server when the report was first created. For example, if a report was created using an ODBC data source, specify the file name "PDSODBC.DLL". For more information on possible database DLLs, refer to Runtime File Requirements online Help.
privateInfo	Specifies the application's handle to the Server connection. In your application, a connection to the Server must already be established before this method is called. Pass the HDBC (handle to a database connection) from this connection to the privateInfo parameter.

Remarks

Use this class method to allow the Crystal Report Engine to "piggyback" your application's existing connection to a Server. This lowers the number of connections established by a workstation, reducing application time and network traffic. It also prevents a LogOff call from disconnecting an application's existing connection to the Server.

The CRPEngine::LogOnSQLServerWithPrivateInfo method can only be used with database connections established by ODBC or Q+E Library 2.0. Any other database connections can not be accessed by this method.

To obtain an HDBC for an ODBC connection, use the following function calls (see the ODBC SDK 2.0 manual for more information):

Function Call	Description
SQLAllocEnv	Initializes the ODBC call level interface and allocates memory for an environment handle.
SQLAllocConnect	Returns an ODBC HDBC. To obtain an HDBC for a Q+E Library connection, use the following function calls (see the InterSolv DataDirect Developer's Toolkit for more information).
qeConnect	Opens a connection to the server.
qeGetODBCHdbc	Returns the ODBC HDBC.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEngine::Open

Syntax

```
Open( ) RETURNING BOOLEAN
```

Remarks

This method opens the Crystal Report Engine. This method is only necessary if you constructed the CRPEngine object with the *open* parameter equal to FALSE. If you set the *open* parameter to TRUE when you constructed the CRPEngine object, this method is unnecessary.

Return Value

Returns TRUE if the Crystal Report Engine was opened successfully, FALSE if something went wrong.

Related Topics

CRPEngine::CRPEngine, Page 961

PFOpenEngine, Page 247

CRPEngine::OpenJob

Syntax

```
OpenJob (reportFileName CHAR(*) ) RETURNING CRPEJob
```

Parameters

Parameter	Description
reportFileName	The name and path (if necessary) of the report file being opened for the specified print job.

Remarks

This method opens the report specified by reportFileName. A pointer to a CRPEJob object is returned. Print job attributes may be referenced through this object.

Return Value

CRPEJob object for the opened print job.

Related Topics

PEDOpenPrintJob, Page 248

CRPEngine::PrintReport

Syntax

```
PrintReport (reportFilePath CHAR (*), toPrinter BOOLEAN : TRUE,  
toWindow BOOLEAN : FALSE, title CHAR (*) : "Crystal Report", left  
SMALLINT : 0,top SMALLINT : 0, width SMALLINT : 0, height SMALLINT : 0,  
style INTEGER : 0, parentWindow ixWindow : NULL) RETURNING SMALLINT
```

Parameters

Parameter	Description
reportFilePath	Specifies the name and path of the report to be printed.
toPrinter	Specifies whether or not the report is to be sent to the default printer.
toWindow	Specifies whether or not the report is to be displayed in a preview window.
title	Specifies the title you want to appear in the title bar if the report is being sent to a preview window.
left	Specifies the x coordinate of the upper left hand corner of the preview window, in device coordinates.
top	Specifies the y coordinate of the upper left hand corner of the preview window, in device coordinates.
width	Specifies the width of the preview window, in device coordinates.
height	Specifies the height of the preview window, in device coordinates.
style	Specifies the style of the window being created. Style setting can be combined using the bitwise "OR" operator. Refer to the CWnd class in the Microsoft Foundation Class Library reference for possible window styles.
parentWindow	Specifies a pointer to the CWnd object for the window that is the parent of the preview window. Specify NULL if the preview window will not have a parent window.

Remarks

This method provides a quick but limited way to print a report. The report may only be output to a printer or preview window. Use this class method any time you simply want to print a report from an application without giving the user the ability to customize the report.

Return Value

0 if the call is successful. Returns an error code if something goes wrong.

Related Topics

PEPrintReport, Page 263

CLASS CRPEJob

In order to open a particular report it is first necessary to have an open Crystal Report Engine object in the application. You may then call *CRPEngine::OpenJob, Page 967*, specifying the report file name to open. If successful, you will be returned a pointer to a CRPEJob object. It is through this object that you may modify the print job attributes as well as output the report in various formats. Almost all of these methods correspond to similar functions available in the Crystal Report Engine API.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEJob	Constructs a CRPEJob object. Used internally by CRPEngine::OpenJob.

CRPEJob::Cancel

Syntax

```
Cancel () RETURNING VOID
```

Remarks

This method cancels processing of a print job. It can be tied to a control that allows the user to cancel a print job in progress.

Related Topics

PECancelPrintJob, Page 121

CRPEJob::CheckFormula

Syntax

```
CheckFormula (*formulaName CHAR (*)) RETURNING BOOLEAN
```

Parameters

Parameter	Description
formulaName	The name of the formula that you need to check for errors.

Remarks

Use this method to check a named formula for errors. This method works just like the Check button in the Formula Editor. If the named formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is correct, FALSE if there is an error in the formula or something goes wrong.

Related Topics

PECheckFormula, Page 123

CRPEJob::CheckGroupSelectionFormula

Syntax

```
CheckGroupSelectionFormula () RETURNING BOOLEAN
```

Remarks

Use this method to check the group selection formula for the report for errors. This method works just like the Check button in the Formula Editor. If the group selection formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is correct, FALSE if there is an error in the formula or something goes wrong.

Related Topics

CRPEJob::CheckSelectionFormula

Syntax

```
CheckSelectionFormula () RETURNING BOOLEAN
```

Remarks

Use this method to check the record selection formula for the report for errors. This method works just like the Check button in the Formula Editor. If the selection formula contains an error, the method returns FALSE.

Return Value

TRUE if the formula text is okay, FALSE if there is an error in the formula or something goes wrong.

Related Topics

PECheckSelectionFormula, Page 124

CRPEJob::Close

Syntax

```
Close () RETURNING VOID
```

Remarks

This method closes the print job. It also calls the destructor for the CRPEJob object. If printing has not yet finished when this method is called, it continues until the job is completely printed. If the preview window is open, it stays open.

Related Topics

PEClosePrintJob, Page 129

CRPEJob::CloseWindow

Syntax

```
CloseWindow () RETURNING VOID
```

Remarks

This method closes the preview window. If you are customizing preview window controls, implement this method to allow the user to close the preview window when finished viewing the report.

Related Topics

PECloseWindow, Page 131

CRPEJob::CRPEJob

Syntax

```
CRPEJob ( jobHandle SMALLINT ) ;
```

Parameters

Parameter	Description
jobHandle	Handle to the job created by CRPEngine::OpenJob().

Remarks

This is the constructor for the class. It also stores the job number internally for future use in Crystal Report Engine API calls.

NOTE: This class constructor is automatically called by CRPEngine::OpenJob(). Do not call this constructor from within your own code.

CRPEJob::DeleteNthGroupSortField

Syntax

```
DeleteNthGroupSortField ( sortFieldN SMALLINT ) RETURNING BOOLEAN
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the group sort field you want to delete. The first group sort field added to the report is field 0, the second is 1, etc.

Remarks

This method deletes the group sort field at the specified position. The group and group summary are not removed from the report, but the field is removed from the list of group sort fields, and the summary data appearing in the group field is no longer sorted.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEDeleteNthGroupSortField, Page 136

CRPEJob::DeleteNthSortField

Syntax

```
DeleteNthSortField (sortFieldN SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the sort field you want to delete. The first sort field added to the report is field 0, the second is 1, etc.

Remarks

This method deletes the specified sort field from the report. The field is not deleted from the report, but it is removed from the list of sort fields, and data in that field no longer appears sorted.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEDeleteNthSortField, Page 138

CRPEJob::ExportPrintWindow

Syntax

```
ExportPrintWindow ( toMail BOOLEAN) RETURNING BOOLEAN
```

Parameters

Parameter	Description
<i>toMail</i>	Specifies whether or not the report file should be exported to an e-mail address. If TRUE, the file is exported to e-mail. If FALSE, the file is exported to a disk file.

Remarks

This method exports the report displayed in the preview window to a disk file or e-mail address. If you are customizing preview window controls, use this method to enable the user to preview the report in the preview window, and if everything looks satisfactory, to export the report to a disk file or e-mail address (in response to a user event - button click, menu command, etc.).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEJob::ExportTo

Syntax

```
ExportTo ( options CRPEExportOptions) RETURNING BOOLEAN
```

Parameters

Parameter	Description
<i>options</i>	A pointer to a CRPEExportOptionsclass.

Remarks

This method sets the output of the print job to be exported. The export format is specified through the *options* parameter. This method does not export the report, but specifies that when the report is printed, it will be exported to a disk file or e-mail address according to settings in the *options* parameter. To actually export the report, use CRPEJob::Start.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEExportTo, Page 146

CRPEJob::GetErrorCode

Syntax

```
GetErrorCode () RETURNING SMALLINT
```

Remarks

This method gets the error code for the print job. When a call to another function fails, this call gets the error code that was generated so you can take some action based on that error code.

Return Value

A Class Library error code. Returns 0 (PEP_NOERROR) if no error has occurred.

Related Topics

PEGetErrorCode, Page 152

CRPEJob::GetErrorText

Syntax

```
GetErrorText () RETURNING ixString
```

Remarks

This method returns a descriptive error message for the print job.

Return Value

Value	Description
ixString	A text description of the current Class Library error if an error has occurred.

Related Topics

PEGGetErrorText, Page 153

CRPEJob::GetExportOptions

Syntax

```
GetExportOptions () RETURNING BOOLEAN, CRPEExportOptions
```

Remarks

This method prompts the user with a series of dialog boxes to specify the export options to be used. These options are used by the Class Library to fill in a CRPEExportOptions class. The CRPEJob::ExportTo function can then be used to set the print job destination using the information in this class.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPEExportOptions	CRPEExportOptions class.

Related Topics

PEGGetExportOptions, Page 155

CRPEJob::GetFormula

Syntax

```
GetFormula (formulaName CHAR(*)) RETURNING BOOLEAN, ixString
```

Parameters

Parameter	Description
formulaName	The name of the formula for which you want to retrieve the formula string.

Remarks

This method returns the formula text for the specified formula. CRPEJob::GetFormula is often used with *CRPEJob::SetFormula, Page 1016*, to identify and then change an existing formula at print time in response to a user selection.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if the named formula does not exist in the report.
ixString	The object passed here is loaded with the specified formula text when CRPEJob::GetFormula completes successfully.

Related Topics

PEGGetFormula, Page 156

CRPEJob::GetGraphData

Syntax

```
GetGraphData (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,  
CRPEGraphDataInfo
```

Parameters

Parameter	Description
sectionCode	Specifies the section of the report containing the graph you want to get information from. Possible values are:

Value	Meaning
PEP_ALLSECTIONS	Graph appears in all sections.

<i>Value</i>	<i>Meaning</i>
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to get graph data information from. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.

Remarks

Use this method to find out what data in your report is being used by a specified graph when the graph is created. This information includes which groups are used to create the rows and columns of the graph and which summary field in the report is used to set the values of the risers in the graph.

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPEGraphDataInfo	A pointer to a CRPEGraphDataInfo class.

Related Topics

PEGGetGraphData, Page 158

CRPEJob::GetGraphOptions

Syntax

```
GetGraphOptions (sectionCode SMALLINT, graphN SMALLINT) RETURNING
BOOLEAN, CRPEGraphOptions
```

Parameters

<i>Parameter</i>	<i>Description</i>																
sectionCode	Specifies the section of the report containing the graph you want to get graph display options for. Possible values are: <table border="1"> <thead> <tr> <th><i>Value</i></th><th><i>Meaning</i></th></tr> </thead> <tbody> <tr> <td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr> <tr> <td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr> <tr> <td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr> <tr> <td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr> <tr> <td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr> <tr> <td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr> <tr> <td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr> </tbody> </table>	<i>Value</i>	<i>Meaning</i>	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
<i>Value</i>	<i>Meaning</i>																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to get graph display options for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.

Remarks

Use this method to obtain information about any of several graph options. These options include the minimum and maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

Return Value

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
graphOptions	CRPEGraphOptions class.

Related Topics

PEGetGraphOptions, Page 159

CRPEJob::GetGraphText

Syntax

```
GetGraphText (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,  
CRPEGraphTextInfo
```

Parameters

Parameter	Description
sectionCode	Specifies the section of the report containing the graph you want to get title text for. Possible values are:

Value	Meaning
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

Parameter	Description
graphN	Specifies which graph within the section you want to get title text for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.

Remarks

This method allows you to find out what text appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs).

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPEGraphTextInfo	CRPEGraphTextInfoclass.

Related Topics

PEGetGraphText, Page 161

CRPEJob::GetGraphType

Syntax

```
GetGraphType (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,  
SMALLINT
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to find out the type of. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.

<i>Value</i>	<i>Meaning</i>
PEP_FOOTERSECTION	Graph appears in the Page Footer section.
<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to know the type of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.

Remarks

Identifies one of the available graph/chart types used for the specified graph in the specified section. Use this method to find out what type of graph is being displayed in the report. There are many types of graphs and charts possible with Seagate Crystal Reports. This method will return the type of graph or chart being displayed.

Return Value

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves a value indicating the type of graph that the specified graph is. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_SIDE_BY_SIDE_BAR_GRAPH	Side By Side bar graph.
PEP_STACKED_BAR_GRAPH	Stacked bar graph.
PEP_PERCENT_BAR_GRAPH	Percent bar graph.
PEP_FADED_3D_SIDE_BY_SIDE_BAR_GRAPH	3D Side By Side bar graph.
PEP_FADED_3D_STACKED_BAR_GRAPH	3D Stacked bar graph.
PEP_FADED_3D_PERCENT_BAR_GRAPH	3D Percent bar graph.
PEP_PIE_GRAPH	Pie graph.
PEP_MULTIPLE_PIE_GRAPH	Multiple Pie graph.

<i>Value</i>	<i>Meaning</i>
PEP_PROPORIONAL _MULTI_PIE_GRAPH	Weighted Pie graph.
PEP_LINE_GRAPH	Line graph.
PEP_AREA_GRAPH	Area graph.
PEP_THREED_BAR_GRAPH	3D bar graph.
PEP_USER_DEFINED _GRAPH	User Defined graph type.
PEP_UNKNOWN_TYPE _GRAPH	Unknown graph type.

Related Topics

PEGGetGraphType, Page 162

CRPEJob::GetGroupCondition

Syntax

```
GetGroupCondition (sectionCode SMALLINT) RETURNING BOOLEAN, ixString,
SMALLINT, SMALLINT
```

Parameter

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the report section for which you want to get the grouping condition. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_GROUPHEADER	Sets the group condition for the Group Header section.
PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.

Remarks

Determines the group condition information for a selected group section in the specified report. Use this method to find out the group condition for a group section. Use *PESetGroupCondition, Page 296* to change the group condition once it is known.

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Specifies the name of the group summary field for which you want to get the grouping condition.
SMALLINT	Retrieves the type of field being used as the condition field and the condition that creates a new group. Use one of the following masks to separate the condition type from the group condition in the condition parameter:

<i>Value</i>	<i>Meaning</i>
PEP_GC_CONDITIONMASK	Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain the group condition value.
PEP_GC_TYPEMASK	Obtains the type of field used for the group condition. Use the bitwise AND (&) to combine this mask with the value of the <i>condition</i> parameter to obtain a value representing the type of field used by the group condition.

For group condition field types other than Date and Boolean, the group condition value of the *condition* parameter is PEP_GC_ANYCHANGE.

<i>Value (Date Fields)</i>	<i>Meaning</i>
PEP_GC_DAILY	Triggers a grouping every time the date changes.
PEP_GC_WEEKLY	Triggers a grouping every time the date changes from one week to the next (a week runs from Sunday through Saturday).
PEP_GC_BIWEEKLY	Triggers a grouping every time the date changes from one two-week period to the next.

<i>Value (Date Fields)</i>	<i>Meaning</i>
PEP_GC_SEMIMONTHLY	Triggers a grouping every time the date changes from one half-month period to the next.
PEP_GC_MONTHLY	Triggers a grouping every time the date changes from one month to the next.
PEP_GC_QUARTERLY	Triggers a grouping every time the date changes from one calendar quarter to the next.
PEP_GC_SEMIANNUALLY	Triggers a grouping every time the date changes from one half-year period to the next.
PEP_GC_ANNUALLY	Triggers a grouping every time the date changes from one year to the next.

<i>Value (Boolean Fields)</i>	<i>Meaning</i>
PEP_GC_TOYES	Triggers a grouping every time the sort and group by field changes from No to Yes.
PEP_GC_TONO	Triggers a grouping every time the sort and group by field changes from Yes to No.
PEP_GC_EVERYYES	Triggers a grouping every time the group and sort by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the group and sort by field value is No.
PEP_GC_NEXTISYES	Triggers a grouping every time the next value in the sort and group by field is Yes.
PEP_GC_NEXTISNO	Triggers a grouping every time the next value in the sort and group by field is No.

The group condition field type portion of the condition parameter will be one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_GC_TYPEOTHER	Any field type other than Date or Boolean. The group condition portion of the <i>condition</i> parameter will be PEP_GC_ANYCHANGE.
PEP_GC_TYPEDATE	A Date field is used to create the group summary field.
PEP_GC_TYPEBOOLEAN	A Boolean field is used to create the group summary field.

<i>Parameter</i>	<i>Description</i>
SMALLINT	Obtains the sort direction for the group summary field. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts data in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts data in descending order (Z to A, 9 to 1).

Related Topics

PEGetGroupCondition, Page 164

CRPEJob::GetGroupSelectionFormula

Syntax

```
GetGroupSelectionFormula () RETURNING BOOLEAN, ixString
```

Remarks

This method returns the formula text for the group selection formula.

CRPEJob::GetGroupSelectionFormula is often used with *CRPEJob::SetGroupSelectionFormula, Page 1025*, to identify and then change an existing group selection formula at print time in response to a user selection.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Specifies the existing group selection formula for the report.

Related Topics

PEGetGroupSelectionFormula, Page 168

CRPEJob::GetJobHandle

Syntax

```
GetJobHandle () RETURNING SMALLINT
```

Remarks

This method returns the job handle for the print job.

Return Value

The print job handle for the CRPEJob object.

CRPEJob::GetJobStatus

Syntax

```
GetJobStatus () RETURNING SMALLINT, CRPEJobInfo
```

Remarks

This method gets the current status of the print job. You can use it in a number of programming situations, for example:

- to trigger error messages, (when a print job fails due to insufficient memory, insufficient disk space, etc.),
- to trigger screen displays (hourglass, series of graphics, etc.) that confirm to the user that work is in progress, or
- to find out whether a job was cancelled by the user after CRPEJob::Start was called.

Return Values

<i>Value</i>	<i>Description</i>														
SMALLINT	Returns one of the following values according to the status of the current print job: <table border="1"><thead><tr><th><i>Value</i></th><th><i>Meaning</i></th></tr></thead><tbody><tr><td>-1</td><td>The Crystal Report Engine has not been opened, or a print job has not been established.</td></tr><tr><td>PEP_JOBNOTSTARTED</td><td>The job has not been started.</td></tr><tr><td>PEP_JOBINPROGRESS</td><td>The job is currently in progress.</td></tr><tr><td>PEP_JOBCOMPLETED</td><td>The job has completed successfully.</td></tr><tr><td>PEP_JOBFAILED</td><td>The job has failed.</td></tr><tr><td>PEP_JOBCANCELLED</td><td>The job has been cancelled by the user.</td></tr></tbody></table>	<i>Value</i>	<i>Meaning</i>	-1	The Crystal Report Engine has not been opened, or a print job has not been established.	PEP_JOBNOTSTARTED	The job has not been started.	PEP_JOBINPROGRESS	The job is currently in progress.	PEP_JOBCOMPLETED	The job has completed successfully.	PEP_JOBFAILED	The job has failed.	PEP_JOBCANCELLED	The job has been cancelled by the user.
<i>Value</i>	<i>Meaning</i>														
-1	The Crystal Report Engine has not been opened, or a print job has not been established.														
PEP_JOBNOTSTARTED	The job has not been started.														
PEP_JOBINPROGRESS	The job is currently in progress.														
PEP_JOBCOMPLETED	The job has completed successfully.														
PEP_JOBFAILED	The job has failed.														
PEP_JOBCANCELLED	The job has been cancelled by the user.														
CRPEJobInfo	Class CRPEJobInfo.														

Related Topics

PEGetJobStatus, Page 171

CRPEJob::GetLineHeight

Syntax

```
GetLineHeight (sectionCode SMALLINT, lineN SMALLINT) RETURNING BOOLEAN,  
SMALLINT, SMALLINT
```

Parameters

<i>Parameter</i>	<i>Description</i>
<i>sectionCode</i>	Specifies the code for the report section containing the line you want to retrieve the height of. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Gets the line height for all sections.
PEP_HEADERSECTION	Gets the line height for the Page Header section.
PEP_GROUPHEADER	Gets the line height for the Group Header section.
PEP_DETAILSECTION	Gets the line height for the Details section.
PEP_GROUPFOOTER	Gets the line height for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the line height for the Grand Total section.
PEP_FOOTERSECTION	Gets the line height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
lineN	Specifies the line number within the specified section for which you want to retrieve the height of. The first line in a section is line 0, the second is 1, etc. Use PEP_ALLLINES to retrieve information on all lines in a section.

Remarks

Gets line height and ascent information for a specified line in a selected section of the report. You can change and pass back a new line height and ascent using *CRPEJob::SetLineHeight*, *Page 1025*.

Return Value

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves the height, in twips, of the specified line.
SMALLINT	Retrieves the line ascent, in twips, of the specified line. Line ascent is the distance from the baseline of the font to the top of the line space.

CRPEJob::GetMargins

Syntax

```
GetMargins () RETURNING BOOLEAN, SMALLINT, SMALLINT, SMALLINT, SMALLINT
```

Remarks

Retrieves the page margin settings for the specified report. Use this method to find out what the currently set margins for the report are. Use [CRPEJob::SetMargins, Page 1026](#), to change report margins.

Return Value

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves the current setting of the left margin in twips.
SMALLINT	Retrieves the current setting of the right margin in twips.
SMALLINT	Retrieves the current setting of the top margin in twips.
SMALLINT	Retrieves the current setting of the bottom margin in twips.

Related Topics

[PEGetMargins, Page 173](#)

CRPEJob::GetMinimumSectionHeight

Syntax

```
GetMinimumSectionHeight (sectionCode SMALLINT) RETURNING BOOLEAN,  
SMALLINT
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the code for the report section for which you want to retrieve the minimum height. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Gets the minimum section height for all sections.
PEP_HEADERSECTION	Gets the minimum section height for the Page Header section.

<i>Value</i>	<i>Meaning</i>
PEP_GROUPHEADER	Gets the minimum section height for the Group Header section.
PEP_DETAILSECTION	Gets the minimum section height for the Details section.
PEP_GROUPFOOTER	Gets the minimum section height for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the minimum section height for the Grand Total section.
PEP_FOOTERSECTION	Gets the minimum section height for the Page Footer section.

Remarks

Retrieves minimum section height information for selected sections in the specified report. Use this method to fetch the minimum section height and pass back using *CRPEJob::SetMinimumSectionHeight, Page 1027*.

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves the minimum height, in twips, for the specified report section.

Related Topics

PEGetMinimumSectionHeight, Page 175

CRPEJob::GetNDetailCopies

Syntax

```
GetNDetailCopies () RETURNING BOOLEAN, SMALLINT
```

Remarks

Returns the number of copies of each Details section in the report that are to be printed. Use this method to find out how many times each Details section of the report will be printed. To change the number of times each Details section is printed, use *CRPEJob::SetNDetailCopies*, Page 1028.

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
nCopies	Retrieves the current setting for the number of times the Details section of the report will be printed.

Related Topics

PEGetNDetailCopies, Page 176

CRPEJob::GetNFormulas

Syntax

```
GetNFormulas () RETURNING SMALLINT
```

Remarks

Use this function to fetch the number of formulas in the report. To fetch the formula by number, use *CRPEJob::GetNthFormula*, Page 996.

Return Value

The number of formulas in the report. If no formulas exist in the report, 0 is returned. If an error occurs, -1 is returned.

Related Topics

PEGetNFormulas, Page 177

CRPEJob::GetNGroups

Syntax

```
GetNGroups () RETURNING SMALLINT
```

Remarks

Use this method to fetch the number of group sections in the report.

Return Values

The number of group sections in the report. Returns -1 if an error occurs.

Related Topics

PEGetNGroups, Page 179

CRPEJob::GetNGroupSortFields

Syntax

```
GetNGroupSortFields () RETURNING SMALLINT
```

Remarks

Returns the number of group sort fields in the specified report. This method is typically used as one of a series: GetNGroupSortFields (called once), *PEGetNthSortField, Page 200* (called as many times as needed to identify the correct group sort field), and *PESetNthGroupSortField, Page 311*, (called once when the correct group sort field is identified). The series can be used to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

Return Value

The number of group sort fields in the report. Returns 0 if there are no group sort fields. Returns -1 if an error occurs.

Related Topics

PEGetNGroupSortFields, Page 180

CRPEJob::GetNLinesInSection

Syntax

```
GetNLinesInSection (sectionCode SMALLINT) RETURNING SMALLINT
```

Parameters

Parameter	Description
Value	Meaning
sectionCode	Specifies the code for the report section that you need to know the number of lines for. Possible values are:
PEP_ALLSECTIONS	Gets the number of lines in all sections.
PEP_HEADERSECTION	Gets the number of lines in the Page Header section.
PEP_GROUPHEADER	Gets the number of lines in the Group Header section.
PEP_DETAILSECTION	Gets the number of lines in the Details section.
PEP_GROUPFOOTER	Gets the number of lines in the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the number of lines in the Grand Total section.
PEP_FOOTERSECTION	Gets the number of lines in the Page Footer section.

Remarks

Use this method to find out how long a specified report section is according to the number of lines in the section. This is useful when trying to fit mailing label reports into the labels you are using, for example.

Return Value

The number of lines in the specified section of the report. Returns -1 if an error occurs.

CRPEJob::GetNParams

Syntax

```
GetNParams () RETURNING SMALLINT
```

Remarks

Use this method whenever you need to know how many parameters are required by a stored procedure in a SQL database table. This method is usually used in conjunction with *CRPEJob::GetNthParam*, *Page 997*, or *CRPEJob::SetNthParam*, *Page 1030*.

Return Value

The number of parameters in the current stored procedure being used to generate the report.

Related Topics

PEGetNParams, *Page 184*

CRPEJob::GetNSortFields

Syntax

```
GetNSortFields () RETURNING SMALLINT
```

Remarks

This method gets the number of sort fields in the report.

Return Value

The number of sort fields defined in the report. Returns 0 (zero) if there are no sort fields in the report. Returns -1 if an error occurs.

Related Topics

PEGetNSortFields, *Page 187*

CRPEJob::GetNTables

Syntax

```
GetNTables() RETURNING SMALLINT
```

Remarks

This method returns the number of tables in the report.

Return Value

The number of tables used in the report. Returns -1 if an error occurs.

CRPEJob::GetNthFormula

Syntax

```
GetNthFormula (formulaN SMALLINT) RETURNING BOOLEAN, ixString, ixString
```

Parameter

Parameter	Description
formulaN	Specifies the number of the formula about which you want to gather information. The first formula added to your report is 0, the second is 1, etc.

Remarks

Use this function to obtain the formula name and formula text of a specific formula in the report. Once the formula name is obtained, formula text can be changed with *CRPEJob::SetFormula*, [Page 1016](#).

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Retrieves the name of the formula specified.
ixString	Retrieves the text of the formula specified.

Related Topics

PEGetNthFormula, [Page 191](#)

CRPEJob::GetNthGroupSortField

Syntax

```
GetNthGroupSortField (sortFieldN SMALLINT) RETURNING BOOLEAN, ixString,  
SMALLINT
```

Parameter

Parameter	Description
sortFieldN	Specifies the number of the group sort field you want to retrieve. The first group sort field is field 0. If the report has N sort fields, the function can be called with sortFieldN between 0 and N-1.

Remarks

Returns information about one of the group sort fields in the specified report: that is, it returns the name of the field and the direction (ascending or descending) of the sort. This method is typically used as one of a series: *CRPEJob::GetNGroupSortFields*, *Page 993* (called once), *CRPEJob::GetNthGroupSortField* (called as many times as needed to identify the correct group sort field), and *CRPEJob::SetNthGroupSortField*, *Page 1029* (called once when the correct sort field is identified). The series can be used to identify and then change an existing group sort field and/or sort order at print time in response to a user selection.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Retrieves the name of the group sort field.
SMALLINT	Retrieves a value indicating the sort direction. Possible values are:

Value	Meaning
PE_SF_ASCENDING	Sorted in Ascending order (A to Z, 1 to 9).
PE_SF_DESCENDING	Sorted in Descending order (Z to A, 9 to 1).

Related Topics

PEGetNthGroupSortField, *Page 193*

CRPEJob::GetNthParam

Syntax

```
GetNthParam (paramN SMALLINT) RETURNING BOOLEAN, ixString
```

Parameter

Parameter	Description
paramN	Specifies which parameter in the stored procedure you want to get the value of. The first parameter of a stored procedure is 0, the second is 1, etc.

Remarks

Gets the Nth parameter of a stored procedure. Use this method whenever you need to find out a particular parameter required by a stored procedure in a SQL database table.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Retrieves the current value of the specified parameter in the stored procedure.

Related Topics

PEGetNthParam, Page 195

CRPEJob::GetNthSortField

Syntax

```
GetNthSortField (sortFieldN SMALLINT) RETURNING BOOLEAN, ixString,  
SMALLINT
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the sort field you want to retrieve. The first sort field added to the report is field 0, the second is 1, etc.

Remarks

This method gets the name of the sort field at the specified sort field position and the direction in which data is sorted (ascending or descending).

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Assigned the name of the sort field if the call completes successfully.
SMALLINT	Assigned the sort direction of the sort field if the call completes successfully. The following values are possible:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorted in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorted in descending order (Z to A, 9 to 1).

Related Topics

PEGetNthSortField, Page 200

CRPEJob::GetNthTableLocation

Syntax

```
GetNthTableLocation (tableN SMALLINT) RETURNING BOOLEAN,  
CRPETableLocation
```

Parameters

<i>Parameter</i>	<i>Description</i>
tableN	Specifies the 0 indexed number of the table for which you want to retrieve location information.

Remarks

This method gets table location information for the specified table in the report. This method is typically combined with the CRPEJob::SetNthTableLocation method to identify the location of a table and then to change it.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if anything goes wrong.
CRPETableLocation	CRPETableLocation class.

Related Topics

PEGetNthTableLocation, Page 203

CRPEJob::GetNthTableLogOnInfo

Syntax

```
GetNthTableLogonInfo (tableN SMALLINT) RETURNING BOOLEAN, CRPELogonInfo
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table from which you want to get log on information.

Remarks

This method gets SQL connection information for the specified table.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if anything goes wrong.
logonInfo	CRPELogOnInfo class.

Related Topics

PEGetNthTableLogOnInfo, Page 205

CRPEJob::GetNthTableSessionInfo

Syntax

```
GetNthTableSessionInfo (tableN SMALLINT) RETURNING BOOLEAN,  
CRPESessionInfo
```

Parameters

Parameter	Description
tableN	A 0 indexed table number indicating which MS Access table in the report you wish to obtain the session information for.

Remarks

This method gets session information for the specified Microsoft Access table. Many MS Access database tables require that a session be opened before the information in the table can be used. Use this method to obtain the session information (User ID and Session Handle) for a particular tab.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if anything goes wrong.
CRPESessionInfo	Class CRPESessionInfo.

Related Topics

PEGetNthTableSessionInfo, Page 206

CRPEJob::GetNthTableType

Syntax

```
GetNthTableType (tableN SMALLINT) RETURNING BOOLEAN, CRPETableType
```

Parameter

Parameter	Description
tableN	0 indexed table number indicating which table in the report for which you want to determine the type.

Remarks

This method gets table type information for the specified table.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if anything goes wrong.
CRPETableType	Class CRPETableType.

Related Topics

PEGetNthTableType, Page 207

CRPEJob::GetPrintDate

Syntax

```
GetPrintDate () RETURNING BOOLEAN, SMALLINT, SMALLINT, SMALLINT
```

Remarks

Determines the print date (if any) that was specified with the report. Use this method to fetch the print date and pass back using *CRPEJob::SetPrintDate, Page 1034*.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves the year for the current print date.
SMALLINT	Retrieves the month for the current print date.
SMALLINT	Retrieves the day for the current print date.

Related Topics

PEGetPrintDate, Page 209

CRPEJob::GetPrintOptions

Syntax

```
GetPrintOptions () RETURNING BOOLEAN, CRPEPrintOptions
```

Remarks

Retrieves the print options specified for the report (the options that are set in the Print Setup common dialog box) and uses them to fill in the CRPEPrintOptions class. Use this function to fetch print options from the report in order to update them and pass back using *CRPEJob::SetPrintOptions, Page 1035*.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPEPrintOptions	A pointer to a CRPEPrintOptions class.

Related Topics

PEGetPrintOptions, Page 211

CRPEJob::GetReportTitle

Syntax

```
GetReportTitle () RETURNING BOOLEAN, ixString
```

Remarks

This method gets the report title for the print job.

Return Values

TRUE if the call is successful, FALSE if something goes wrong.

Value	Description
ixString	Assigned the title of the report.

Related Topics

PEGetReportTitle, Page 212

CRPEJob::GetSectionFormat

Syntax

```
GetSectionFormat (sectionCode SMALLINT) RETURNING BOOLEAN,  
CRPESectionOptions
```

Parameter

Parameter	Description
sectionCode	Specifies the code for the report section that you need to obtain format information for. Possible values are:

Value	Meaning
PEP_ALLSECTIONS	Gets the format information for all sections.
PEP_HEADERSECTION	Gets the format information for the Page Header section.
PEP_GROUPHEADER	Gets the format information for the Group Header section.
PEP_DETAILSECTION	Gets the format information for the Details section.
PEP_GROUPFOOTER	Gets the format information for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the format information for the Grand Total section.
PEP_FOOTERSECTION	Gets the format information for the Page Footer section.

Remarks

Retrieves the section format settings for a selected section in the specified report and supplies the information by assigning values to the members of the CRPESectionOptions class. Use this method in order to edit and update the section formats and pass information back using *CRPEJob::SetSectionFormat, Page 1036*.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPESectionOptions	CRPESectionOptions class.

Related Topics

CRPEJob::GetSectionFormat

Syntax

```
GetSectionFormat (sectionCode SMALLINT) RETURNING BOOLEAN,
CRPESectionOptions
```

Parameter

Parameter	Description
sectionCode	Specifies the code for the report section that you need to obtain format information for. Possible values are:
Value	Meaning
PEP_ALLSECTIONS	Gets the format information for all sections.
PEP_HEADERSECTION	Gets the format information for the Page Header section.
PEP_GROUPHEADER	Gets the format information for the Group Header section.
PEP_DETAILSECTION	Gets the format information for the Details section.
PEP_GROUPFOOTER	Gets the format information for the Group Footer section.
PEP_GRANDTOTALSECTION	Gets the format information for the Grand Total section.
PEP_FOOTERSECTION	Gets the format information for the Page Footer section.

Remarks

Retrieves the section format settings for a selected section in the specified report and supplies the information by assigning values to the members of the CRPESectionOptions class. Use this method in order to edit and update the section formats and pass information back using *CRPEJob::SetSectionFormat*, Page 1036.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
CRPESectionOptions	CRPESectionOptions class.

Related Topics

PEGetSectionFormat, Page 217

CRPEJob::GetSelectedPrinter

Syntax

```
GetSelectedPrinter () RETURNING BOOLEAN, ixString, ixString, ixString,  
FOREIGN
```

Remarks

If a non-default printer is specified in the report, this method retrieves information about that printer. If you need to know which printer has been specified with the report, use CRPEJob::GetSelectedPrinter to obtain the information. This can be useful to determine if the user has access to the printer specified in the report and to choose another printer if not.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Retrieves the name of the printer driver for the currently selected printer in the report.
ixString	Retrieves the name of the printer currently selected in the report.
ixString	Retrieves the name of the port the currently selected printer is attached to. For example, "LPT1:".
FOREIGN	A DEVMODE class that contains information on the currently selected printer, if the CRPEJob::GetSelectedPrinter method completes successfully. For more information, see <i>DEVMODE, Page 353</i> .

Related Topics

CRPEJob::GetSelectionFormula

Syntax

```
GetSelectionFormula() RETURNING BOOLEAN, ixString
```

Remarks

This method returns the formula text for the record selection formula. CRPEJob::GetSelectionFormula is often used with *CRPEJob::GetSelectionFormula, Page 1007*, to identify and then change an existing selection formula at print time in response to a user selection.

Return Values

<i>Value</i>	<i>Description</i>
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Specifies the existing selection formula for the report.

Related Topics

PEGetSelectionFormula, Page 224

CRPEJob::GetSQLQuery

Syntax

```
GetSQLQuery () RETURNING BOOLEAN, ixString
```

Remarks

This method retrieves the SQL query that will be sent to the database server. This method can be used with CRPEJob::SetSQLQuery to retrieve and then change the SQL query for the report.

Return Values

TRUE if the call is successful, FALSE if something goes wrong.

<i>Value</i>	<i>Description</i>
ixString	Assigned the text of the SQL query being sent to the server if the call completes successfully.

Related Topics

PEGetSQLQuery, Page 226

CRPEJob::IsJobFinished

Syntax

```
IsJobFinished () RETURNING BOOLEAN
```

Remarks

This method returns a Boolean value that indicates whether or not processing has finished for the print job. You can use this method any time you have a call that is contingent on a print job being finished.

Return Value

TRUE if processing has finished, FALSE if the job is in progress.

Related Topics

PEIsPrintJobFinished, Page 238

CRPEJob::NextWindowMagnification

Syntax

```
NextWindowMagnification () RETURNING BOOLEAN
```

Remarks

This method switches to the next preview window magnification in order. Use this function to cycle through the three levels of preview window magnification whenever the report has been printed to a preview window. The three levels of magnification are: Full Page, Fit One Side, and Fit Both Sides.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PENextPrintWindowMagnification, Page 245

CRPEJob::OutputToPrinter

Syntax

```
OutputToPrinter (nCopies SMALLINT : 1) RETURNING BOOLEAN
```

Parameters

Parameter	Description
nCopies	Specifies how many copies of the report are to be printed. Default is 1 copy.

Remarks

This method sets the output of the print job to the printer with the specified number of copies. This method does not print the report, but specifies that when the report is printed, it will be sent to a printer. To actually print the report, use CRPEJob::Start.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

CRPEJob::OutputToWindow

Syntax

```
OutputToWindow (title CHAR (*), left SMALLINT, top SMALLINT, width  
SMALLINT, height SMALLINT, style INTEGER, parentWindow ixWindow)  
RETURNING BOOLEAN
```

Parameters

Parameter	Description
title	Specifies the title you want to appear in the title bar.
left	Specifies the x coordinate of the upper left hand corner of the preview window, in device coordinates.
top	Specifies the y coordinate of the upper left hand corner of the preview window, in device coordinates.
width	Specifies the width of the preview window, in device coordinates.
height	Specifies the height of the preview window, in device coordinates.
style	Specifies the style of the window being created. Style setting can be combined using the bitwise Or operator (). Refer to the CWnd class in the Microsoft Foundation Class Library reference for possible window styles.
parentWindow	Specifies a pointer to the CWnd object for the window that is the parent of the preview window. Specify NULL if the preview window will not have a parent window.

Remarks

This method sets the output of the print job to the preview window which will have the specified attributes. This method does not print the report, but specifies that when the report is printed, it will appear in a preview window. To actually print the report, use CRPEJob::Start.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEOutputToWindow, Page 257

CRPEJob::PrintControlsShowing

Syntax

```
PrintControlsShowing () RETURNING BOOLEAN, SMALLINT
```

Remarks

Checks if the print controls are displayed in the preview window. Use *CRPEJob::ShowPrintControls*, [Page 1040](#), to change whether or not print controls will appear in the preview window.

Return Values

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Returns a pointer to a TRUE value if the print controls will be shown in the preview window, FALSE if they will be hidden.

Related Topics

PEPrintControlsShowing, [Page 261](#)

CRPEJob::PrintWindow

Syntax

```
PrintWindow () RETURNING BOOLEAN
```

Remarks

This method prints the report displayed in the preview window to the printer. If you are customizing preview window controls, use this method to enable the user to preview the report in the preview window, and then, if everything looks satisfactory, to print the report to the printer (in response to a user event - button click, menu command, etc.).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEPrintWindow, [Page 265](#)

CRPEJob::SelectPrinter

Syntax

```
SelectPrinter (driverName CHAR (*), printerName CHAR (*), portName CHAR (*), mode FOREIGN) RETURNING BOOLEAN
```

Parameters

Parameter	Description
driverName	Specifies the name of the printer driver for the printer being selected.
printerName	Specifies the name of the printer being selected (as indicated in the Printers Control Panel).
PortName	Specifies the name of the port the printer is attached to. For example: "LPT1:".
mode	A pointer to a DEVMODE class. The default implementation of CRPEJob::SelectPrinter ignores this parameter. For more information, see <i>DEVMODE, Page 353</i> .

Remarks

This method specifies the printer and/or print characteristics for the print job. You can use this method to enable the user to select a printer other than the default printer at print time. One way of doing this is to have your application call the Windows common Print Setup dialog box.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetPrintDate, Page 324

CRPEJob::SetFont

Syntax

```
SetFont (sectionCode SMALLINT, scopeCode SMALLINT, faceName CHAR (*),  
fontFamily SMALLINT, fontPitch SMALLINT, charSet SMALLINT, pointSize  
SMALLINT, isItalic SMALLINT, isUnderlined SMALLINT, isStruckOut  
SMALLINT, weight SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
sectionCode	Specifies the section of the report for which you want to set the font. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Sets the line height for all sections.
PEP_HEADERSECTION	Sets the line height for the Page Header section.
PEP_GROUPHEADER	Sets the line height for the Group Header section.
PEP_DETAILSECTION	Sets the line height for the Details section.
PEP_GROUPFOOTER	Sets the line height for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the line height for the Grand Total section.
PEP_FOOTERSECTION	Sets the line height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
scopeCode	Specifies whether the font selected is to apply to fields, to text, or to both. To specify both, use the bitwise Or operator: . The following values are possible:

<i>Value</i>	<i>Meaning</i>
PEP_FIELDS	Sets the default font for fields in the report section specified.
PEP_TEXT	Sets the default font for all text (that has not been entered as a text field value) in the report section specified.

<i>Parameter</i>	<i>Description</i>
faceName	Specifies the actual face name of the font you want to use. The face name you pass can typically come from a font dialog box, be hard coded in the application, or be chosen by the application from the fonts supported on the printer. For example: "Times New Roman".
fontFamily	Specifies the font family for the font you want to use. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
FF_DONTCARE	No font family or family does not matter.
FF_ROMAN	Variable pitch font with serifs.
FF_SWISS	Fixed pitch font without serifs.
FF_MODERN	Fixed pitch font, with or without serifs.
FF_SCRIPT	Handwriting-like font.
FF_DECORATIVE	Fancy display font.

<i>Parameter</i>	<i>Description</i>
fontPitch	Specifies the font pitch you wish to use. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
DEFAULT_PITCH	Retains the default pitch for the font.
FIXED_PITCH	Fixed pitch, each character is the same width.
VARIABLE_PITCH	Variable pitch, the width of each character varies.

<i>Parameter</i>	<i>Description</i>
charSet	Specifies the character set you wish to use. Use one of the following values:

<i>Value</i>
ANSI_CHARSET
DEFAULT_CHARSET
SYMBOL_CHARSET
HANGEUL_CHARSET
OEM_CHARSET
SHIFTJIS_CHARSET
CHINESEBIG5_CHARSET

<i>Parameter</i>	<i>Description</i>
pointSize	Specifies the desired point size for the selected font. Use 0 to indicate no change.
isItalic	Specifies whether the font selected should be italicized. Use 1 for italics, 0 for no italics, or PEP_UNCHANGED to leave the italics as set up in the report.
isUnderlined	Specifies whether the font should be underlined. Use 1 to underline, 0 for no underline, or PEP_UNCHANGED to leave underline settings as specified in the report.
isStruckOut	Specifies whether or not the font should appear in strikethrough format. Use 1 for strike-out, 0 for no strike out, or PEP_UNCHANGE to leave strike-out settings as specified in the report.
weight	Specifies the weight of the font. Possible values are:

<i>Value</i>
FW_DONTCARE
FW_EXTRALIGHT
FW_NORMAL
FW_SEMIBOLD
FW_EXTRABOLD
FW_ULTRALIGHT
FW_DEMIBOLD
FW_BLACK
FW_THIN
FW_LIGHT
FW_MEDIUM
FW_BOLD
FW_HEAVY
FW_REGULAR
FW_ULTRABOLD

Remarks

This method sets the font and font characteristics for the specified section. Use any time you need to change a default font at runtime in response to user input, or to specify a built-in printer font.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetFont, Page 282

CRPEJob::SetFormula

Syntax

```
SetFormula (formulaName CHAR(*), formulaText CHAR(*)) RETURNING BOOLEAN
```

Parameters

Parameter	Description
formulaName	The name of the formula for which you want to assign new formula text.
formulaText	The actual formula text that replaces the existing formula string.

Remarks

This method sets the formula text for the specified formula. CRPEJob::SetFormula is often used with *CRPEJob::GetFormula, Page 976*, to identify and then change an existing formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if the named formula does not exist in the report, or there is an error in the formula.

Related Topics

PESetFormula, Page 287

CRPEJob::SetGraphData

Syntax

```
SetGraphData (sectionCode SMALLINT, graphN SMALLINT, graphDataInfo  
CRPEGraphDataInfo) RETURNING BOOLEAN
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to change graph data information for. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																

Parameter	Description
graphN	Specifies which graph within the section you want to change graph data information for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphDataInfo	A pointer to a CRPEGraphDataInfoclass.

Remarks

Use this method to change what data is used from your report to create a specified graph. This information includes the groups used to create the rows and columns of the graph and the summary field used to set the values of the risers in the graph.

Return Value

Value	Description
BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

CRPEJob::SetGraphOptions

Syntax

```
SetGraphOptions (sectionCode SMALLINT, graphN SMALLINT, graphOptions
CRPEGraphOptions) RETURNING BOOLEAN
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to change display options for. Possible values are:																
	<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr> <tr> <td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr> <tr> <td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr> <tr> <td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr> <tr> <td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr> <tr> <td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr> <tr> <td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr> </tbody> </table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																
graphN	Specifies which graph within the section you want to change display options for. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.																
graphOptions	A pointer to a CRPEGraphOptions class.																

Remarks

Use this function to change any of several graph options. These options include the minimum and

maximum values that can appear on the graph, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the graph.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphOptions, Page 291

CRPEJob::SetGraphText

Syntax

```
SetGraphText (sectionCode SMALLINT, graphN SMALLINT, graphTextInfo  
CRPEGraphTextInfo) RETURNING BOOLEAN
```

Parameters

Parameter	Description																
sectionCode	Specifies the section of the report containing the graph you want to change the text of. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Graph appears in all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Graph appears in the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Graph appears in the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Graph appears in the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Graph appears in the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Graph appears in the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Graph appears in the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Graph appears in all sections.	PEP_HEADERSECTION	Graph appears in the Page Header section.	PEP_GROUPHEADER	Graph appears in the Group Header section.	PEP_DETAILSECTION	Graph appears in the Details section.	PEP_GROUPFOOTER	Graph appears in the Group Footer section.	PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.	PEP_FOOTERSECTION	Graph appears in the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Graph appears in all sections.																
PEP_HEADERSECTION	Graph appears in the Page Header section.																
PEP_GROUPHEADER	Graph appears in the Group Header section.																
PEP_DETAILSECTION	Graph appears in the Details section.																
PEP_GROUPFOOTER	Graph appears in the Group Footer section.																
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.																
PEP_FOOTERSECTION	Graph appears in the Page Footer section.																

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to change the text of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphTextInfo	A pointer to a CRPEGraphTextInfo class.

Remarks

This method allows you to set or change the text that appears with a graph. A graph can have a title, subtitle, footnote, title for groups, title for series, title for the X axis, title for the Y axis, and title for the Z axis (in 3D graphs).

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphText, Page 292

CRPEJob::SetGraphType

Syntax

```
SetGraphType (sectionCode SMALLINT, graphN SMALLINT, graphType
SMALLINT) RETURNING BOOLEAN
```

Parameters

<i>Parameter</i>	<i>Description</i>
sectionCode	Specifies the section of the report containing the graph you want to change the type of. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Graph appears in all sections.
PEP_HEADERSECTION	Graph appears in the Page Header section.
PEP_GROUPHEADER	Graph appears in the Group Header section.

<i>Value</i>	<i>Meaning</i>
PEP_DETAILSECTION	Graph appears in the Details section.
PEP_GROUPFOOTER	Graph appears in the Group Footer section.
PEP_GRANDTOTALSECTION	Graph appears in the Grand Total section.
PEP_FOOTERSECTION	Graph appears in the Page Footer section.

<i>Parameter</i>	<i>Description</i>
graphN	Specifies which graph within the section you want to change the type of. This value is 0 indexed. Within a section, graphs are numbered from top to bottom first, then from left to right if they have the same top.
graphType	Specifies the style of the graph you want to set. Possible values are:

<i>Value</i>	<i>Meaning</i>
PEP_SIDE_BY_SIDE_BAR_GRAPH	Side By Side bar graph.
PEP_STACKED_BAR_GRAPH	Stacked bar graph.
PEP_PERCENT_BAR_GRAPH	Percent bar graph.
PEP_FADED_3D_SIDE_BY_SIDE_BAR_GRAPH	3D Side By Side bar graph.
PEP_FADED_3D_STACKED_BAR_GRAPH	3D Stacked bar graph.
PEP_FADED_3D_PERCENT_BAR_GRAPH	3D Percent bar graph.
PEP_PIE_GRAPH	Pie graph.
PEP_MULTIPLE_PIE_GRAPH	Multiple Pie graph.
PEP_PROPORIONAL_MULTI_PIE_GRAPH	Weighted Pie graph.
PEP_LINE_GRAPH	Line graph.
PEP_AREA_GRAPH	Area graph.
PEP_THREED_BAR_GRAPH	3D bar graph.

<i>Value</i>	<i>Meaning</i>
PEP_USER_DEFINED_GRAPH	User Defined graph type.
PEP_UNKNOWN_TYPE_GRAPH	Unknown graph type.

Remarks

Changes the type of graph displayed for the specified graph in the specified section based on one of the available graph/chart types. Use this method to change the type of graph that is displayed in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGraphType, Page 294

CRPEJob::SetGroupCondition

Syntax

```
SetGroupCondition (sectionCode SMALLINT, conditionField CHAR(*),
condition SMALLINT, sortDirection SMALLINT) RETURNING BOOLEAN
```

Parameters

<i>Parameter</i>	<i>Description</i>
<i>sectionCode</i>	Specifies the code for the group section for which you want to set the group condition. Select a section from the table below:

<i>Value</i>	<i>Meaning</i>
PEP_GROUPHEADER	Sets the group condition for the Group Header section.
PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.

<i>Parameter</i>	<i>Description</i>
conditionField	Specifies the field that triggers a summary whenever its value changes. Use the name of the field as indicated in the report file.
condition	Specifies the condition that needs to be met for Date and Boolean fields. For all field types except Date and Boolean, use PEP_GC_ANYCHANGE as the condition parameter.
<i>Value (Date Fields)</i>	<i>Meaning</i>
PEP_GC_DAILY	Triggers a grouping every time the date changes.
PEP_GC_WEEKLY	Triggers a grouping every time the date changes from one week to the next (a week runs from Sunday through Saturday).
PEP_GC_BIWEEKLY	Triggers a grouping every time the date changes from one two-week period to the next.
PEP_GC_SEMIMONTHLY	Triggers a grouping every time the date changes from one half-month period to the next.
PEP_GC_MONTHLY	Triggers a grouping every time the date changes from one month to the next.
PEP_GC_QUARTERLY	Triggers a grouping every time the date changes from one calendar quarter to the next.
PEP_GC_SEMIANNUALLY	Triggers a grouping every time the date changes from one half-year period to the next.
PEP_GC_ANNUALLY	Triggers a grouping every time the date changes from one year to the next.

<i>Value (Boolean Fields)</i>	<i>Meaning</i>
PEP_GC_TOYES	Triggers a grouping every time the sort and group by field changes from No to Yes.
PEP_GC_TONO	Triggers a grouping every time the sort and group by field changes from Yes to No.
PEP_GC_EVERYYES	Triggers a grouping every time the group and sort by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the group and sort by field value is No.
PEP_GC_NEXTISYES	Triggers a grouping every time the next value in the sort and group by field is Yes.
PEP_GC_NEXTISNO	Triggers a grouping every time the next value in the sort and group by field is No.

<i>Parameter</i>	<i>Description</i>
sortDirection	Specifies one of the following sort directions:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts data in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts data in descending order (Z to A, 9 to 1).

Remarks

This method sets the condition of the grouping for the specified group section. This method can only replace the group condition for an existing group. It can not create a new group. Use this function whenever you want to change the grouping at print time, for example, to print one report grouped in several different ways.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetGroupCondition, Page 296

CRPEJob::SetGroupSelectionFormula

Syntax

```
SetGroupSelectionFormula (formulaText CHAR(*) ) RETURNING BOOLEAN
```

Parameters

Parameter	Description
formulaText	Specifies the new group selection formula to be assigned to the report.

Remarks

This method sets the formula text for the group selection formula. CRPEJob::SetGroupSelectionFormula is often used with CRPEJob::GetGroupSelectionFormula to identify and then change an existing group selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong, or there is an error in the formula.

Related Topics

PESetGroupSelectionFormula, Page 303

CRPEJob::SetLineHeight

Syntax

```
SetLineHeight (sectionCode SMALLINT, lineN SMALLINT, height SMALLINT,  
ascent SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
sectionCode	Specifies the section containing the line for which you want to specify the height. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Sets the line height for all sections.
PEP_HEADERSECTION	Sets the line height for the Page Header section.
PEP_GROUPHEADER	Sets the line height for the Group Header section.
PEP_DETAILSECTION	Sets the line height for the Details section.
PEP_GROUPFOOTER	Sets the line height for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the line height for the Grand Total section.
PEP_FOOTERSECTION	Sets the line height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
lineN	Specifies the line for which you are setting the height. The first line in a section is line number 0, the next is 1, etc. Use PEP_ALLLINES to specify all lines in the section.
height	Specifies the height, in twips, of the line.
ascent	Specifies the ascent, in twips, of the line. Line ascent is the distance from the baseline of the font to the top of the line space.

Remarks

This method sets the line height for the specified section. This method can be used for setting up a report to print on preprinted forms. It can also be used if you want to insure the size of a line with relation to the font size, for example, to specify a 12 point line with 10 point type.

Return Value

TRUE if the call is successful, FALSE if you have specified an invalid section or something goes wrong.

CRPEJob::SetMargins

Syntax

```
SetMargins (left SMALLINT, right SMALLINT, top SMALLINT, bottom  
SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
left	Specifies the left margin in twips.
right	Specifies the right margin in twips.
top	Specifies the top margin in twips.
bottom	Specifies the bottom margin in twips.

NOTE: For any of the above parameters, PM_SM_DEFAULT can be used to specify that the report use default printer margins.

Remarks

This method sets the page margins for the print job. Use this method any time you want to allow the user to change margins.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetMargins, Page 305

CRPEJob::SetMinimumSectionHeight

Syntax

```
SetMinimumSectionHeight (sectionCode SMALLINT, height SMALLINT)  
RETURNING BOOLEAN
```

Parameters

Parameter	Description
sectionCode	Specifies the section for which you want to specify the minimum height. Use one of the following values:

<i>Value</i>	<i>Meaning</i>
PEP_ALLSECTIONS	Sets the minimum section height for all sections.
PEP_HEADERSECTION	Sets the minimum section height for the Page Header section.
PEP_GROUPHEADER	Sets the minimum section height for the Group Header section.
PEP_DETAILSECTION	Sets the minimum section height for the Details section.
PEP_GROUPFOOTER	Sets the minimum section height for the Group Footer section.
PEP_GRANDTOTALSECTION	Sets the minimum section height for the Grand Total section.
PEP_FOOTERSECTION	Sets the minimum section height for the Page Footer section.

<i>Parameter</i>	<i>Description</i>
height	Specifies the minimum height in twips of the specified section.

Remarks

This method sets the minimum height for the specified section. For example, use this method whenever you want to specify a minimum section height for printing on pre-printed forms or printing on any other kind of document with a fixed format.

Return Value

TRUE if the call is successful, FALSE if you have specified an invalid section or something goes wrong.

Related Topics

PESetMinimumSectionHeight, Page 307

CRPEJob::SetNDetailCopies

Syntax

```
SetNDetailCopies (nCopies SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
<i>nCopies</i>	Specifies the number of copies of the detail section of the report to be printed.

Remarks

This method sets the number of times the Details section of the report is to be printed. For example, you can use this method to print multiple copies of labels for a customer, multiple copies of a purchase order, or multiple copies of anything set up in the Details section of the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNDetailCopies, Page 309

CRPEJob::SetNthGroupSortField

Syntax

```
SetNthGroupSortField (sortFieldN SMALLINT, field CHAR(*), direction SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
<i>sortFieldN</i>	Specifies the number of the group sort field you want to set. The first group sort field added to the report is field 0, the second is 1, etc. If the report has N group sort fields, the function can be called with this parameter between 0 and N-1 to replace an existing group sort field. Call the function with this parameter equal to N to add a new group sort field.
<i>field</i>	Specifies the name of the group field to be sorted.
<i>direction</i>	Specifies the sort direction. The following values are possible:

<i>Value</i>	<i>Meaning</i>
PEP_SF_ASCENDING	Sorts in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts in descending order (Z to A, 9 to 1).

Remarks

This method sorts the specified group summary field in the specified direction (ascending or descending). This method does not create a new group, but will sort an existing group summary field.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthGroupSortField, Page 311

CRPEJob::SetNthParam

Syntax

```
SetNthParam (paramN SMALLINT, paramValue CHAR (*)) RETURNING BOOLEAN
```

Parameters

<i>Parameter</i>	<i>Description</i>
paramN	Specifies which parameter in the stored procedure you want to set the value of. The first parameter of a stored procedure is 0, the second is 1, etc.
paramValue	Specifies the new value of the indicated parameter. This value must be a string. Please see Remarks below.

Remarks

Sets the value of a parameter in a stored procedure. Use this method when working with stored procedures in SQL database tables to set the value of a parameter in a stored procedure. When passing parameter values, all parameter values must be passed as string values. If you wish to pass a numeric value, pass the value in quotes like this: "100".

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetNthParam, Page 313

CRPEJob::SetNthSortField

Syntax

```
SetNthSortField (sortFieldN SMALLINT, field CHAR(*), direction  
SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
sortFieldN	Specifies the number of the sort field you want to set. The first sort field added to the report is field 0, the second is 1, etc. If the report has N sort fields, the function can be called with this parameter between 0 and N-1 to replace an existing sort field. Call the function with this parameter equal to N to add a new sort field.
field	Specifies the name of the field to be sorted.
direction	Specifies the sort direction. The following values are possible:

Value	Meaning
PEP_SF_ASCENDING	Sorts in ascending order (A to Z, 1 to 9).
PEP_SF_DESCENDING	Sorts in descending order (Z to A, 9 to 1).

Remarks

This method sorts report data according to the specified field and direction.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

CRPEJob::SetNthTableLocation

Syntax

```
SetNthTableLocation (tableN SMALLINT, tableLocation CRPETableLocation)
RETURNS BOOLEAN
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table for which you want to set a new location.
tableLocation	A pointer to a CRPETableLocation class.

Remarks

This method sets table location information for the specified table in the report. This method is typically combined with the *CRPEJob::GetNthTableLocation*, Page 999, method to identify the location of a table and then to change it.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PESetNthTableLocation, Page 319

CRPEJob::SetNthTableLogOnInfo

Syntax

```
SetNthTableLogonInfo (tableN SMALLINT, logonInfo CRPELogOnInfo,
propagate BOOLEAN) RETURNS BOOLEAN
```

Parameters

Parameter	Description
tableN	Specifies the 0 indexed number of the table for which you want to set log on information.
logonInfo	A pointer to a CRPELogOnInfo class.
propagate	TRUE or FALSE value indicating whether the log on information should be used for opening all tables being used in the report.

Remarks

This method sets SQL connection information for the specified table. The propagate flag may be used to cause the change to affect all tables with similar server and database properties.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PESetNthTableLogOnInfo, Page 320

CRPEJob::SetNthTableSessionInfo

Syntax

```
SetNthTableSessionInfo (tableN SMALLINT, sessionInfo CRPESessionInfo,  
propagate BOOLEAN) RETURNING BOOLEAN
```

Parameters

Parameter	Description
tableN	0 indexed table number indicating which MS Access table in the report the session is being opened for.
sessionInfo	A pointer to a CRPESessionInfo class.
propagate	TRUE or FALSE value indicating whether the session information should be used for opening all tables being used in the report.

Remarks

This method sets session information for the specified Microsoft Access table. Many MS Access database tables require that a session be opened before the table can be used. Use this method to open the session.

Return Value

TRUE if the call is successful, FALSE if anything goes wrong.

Related Topics

PESetNthTableSessionInfo, Page 322

CRPEJob::SetPrintDate

Syntax

```
SetPrintDate (v_year SMALLINT, v_month SMALLINT, v_day SMALLINT)
RETURNING BOOLEAN
```

Parameters

Parameter	Description
v_year	Specifies the year of the new print date.
v_month	Specifies the month of the new print date.
v_day	Specifies the day of the new print date.

Remarks

This method sets the print date for the report. This method does not schedule the report to print at a different time or day, but only changes the date that appears in any Print Date Field that appears on the report. Use this method, for example, to post date a report when it is printed before the day it is distributed.

Return Values

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetPrintDate, Page 324

CRPEJob::SetPrintOptions

Syntax

```
SetPrintOptions (options CRPEPrintOptions) RETURNING BOOLEAN
```

Parameters

Parameter	Description
options	A pointer to a CRPEPrintOptions class.

Remarks

This method may be used to set the print characteristics of a print job. Use this method any time you want to set the starting page number, the ending page number, the number of report copies, and/or collation instructions for a print job at runtime in response to user specifications.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetPrintOptions, Page 326

CRPEJob::SetReportTitle

Syntax

```
SetReportTitle (title CHAR(*)) RETURNING BOOLEAN
```

Parameter

Parameter	Description
title	Specifies a new title for the report.

Remarks

This method sets the report title for the print job. Use this method whenever you need to change the title of a report at print time.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetReportTitle, Page 329

CRPEJob::SetSectionFormat

Syntax

```
SetSectionFormat (sectionCode SMALLINT, options CRPESectionOptions)
RETURNS BOOLEAN
```

Parameters

Parameter	Description																
sectionCode	Specifies the code for the report section that you need to change format information for. Possible values are: <table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PEP_ALLSECTIONS</td><td>Sets format information for all sections.</td></tr><tr><td>PEP_HEADERSECTION</td><td>Sets the format information for the Page Header section.</td></tr><tr><td>PEP_GROUPHEADER</td><td>Sets the format information for the Group Header section.</td></tr><tr><td>PEP_DETAILSECTION</td><td>Sets the format information for the Details section.</td></tr><tr><td>PEP_GROUPFOOTER</td><td>Sets the format information for the Group Footer section.</td></tr><tr><td>PEP_GRANDTOTALSECTION</td><td>Sets the format information for the Grand Total section.</td></tr><tr><td>PEP_FOOTERSECTION</td><td>Sets the format information for the Page Footer section.</td></tr></tbody></table>	Value	Meaning	PEP_ALLSECTIONS	Sets format information for all sections.	PEP_HEADERSECTION	Sets the format information for the Page Header section.	PEP_GROUPHEADER	Sets the format information for the Group Header section.	PEP_DETAILSECTION	Sets the format information for the Details section.	PEP_GROUPFOOTER	Sets the format information for the Group Footer section.	PEP_GRANDTOTALSECTION	Sets the format information for the Grand Total section.	PEP_FOOTERSECTION	Sets the format information for the Page Footer section.
Value	Meaning																
PEP_ALLSECTIONS	Sets format information for all sections.																
PEP_HEADERSECTION	Sets the format information for the Page Header section.																
PEP_GROUPHEADER	Sets the format information for the Group Header section.																
PEP_DETAILSECTION	Sets the format information for the Details section.																
PEP_GROUPFOOTER	Sets the format information for the Group Footer section.																
PEP_GRANDTOTALSECTION	Sets the format information for the Grand Total section.																
PEP_FOOTERSECTION	Sets the format information for the Page Footer section.																
options	A pointer to a CRPESectionOptions class.																

Remarks

Sets the section format settings for selected sections in the specified report to the values in the CRPESectionOptions class. This method can be used to provide specialized formatting for printing invoices, form letters, printing to pre-printed forms, etc. It allows you to hide a section, insert a page break either before or after a section begins, reset the page number to 1 after a group value prints, prevent page breaks from spreading data from a single record over two pages, and to print group values only at the bottom of a page. For a complete discussion of each of these options, see *CLASS CRPESectionOptions, Page 1056*.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetSectionFormat, Page 331

CRPEJob::SetSelectionFormula

Syntax

```
SetSelectionFormula (formulaText CHAR( * )) RETURNING BOOLEAN
```

Parameters

Parameter	Description
formulaText	Specifies the new selection formula to be assigned to the report.

Remarks

This method sets the formula text for the record selection formula. CRPEJob::SetSelectionFormula is often used with *CRPEJob::GetSelectionFormula, Page 1007*, to identify and then change an existing selection formula at print time in response to a user selection.

Return Value

TRUE if the call is successful, FALSE if something goes wrong, or there is an error in the formula.

Related Topics

PESetSelectionFormula, Page 336

CRPEJob::SetSQLQuery

Syntax

```
SetSQLQuery (query CHAR(*)) RETURNING BOOLEAN
```

Parameters

Parameter	Description
query	The text of the new SQL query to be sent to the server.

Remarks

This method sets the SQL query that will be sent to the database server. This method can be used with CRPEJob::GetSQLQuery to retrieve and then change the SQL query for the report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PESetSQLQuery, Page 337

CRPEJob::ShowFirstPage

Syntax

```
ShowFirstPage () RETURNING BOOLEAN
```

Remarks

This method shows the first page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

CRPEJob::ShowLastPage

Syntax

```
ShowLastPage () RETURNING BOOLEAN
```

Remarks

This method shows the last page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

CRPEJob::ShowNextPage

Syntax

```
ShowNextPage () RETUNING BOOLEAN
```

Remarks

This method shows the next page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

CRPEJob::ShowPreviousPage

Syntax

```
ShowPreviousPage () RETURNING BOOLEAN
```

Remarks

This method shows the previous page in the preview window. Use this method to customize how a user moves through pages in a report.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::ShowPrintControls

Syntax

```
ShowPrintControls (showControls BOOLEAN) RETURNING BOOLEAN
```

Parameters

Parameter	Description
showControls	TRUE indicates default Print controls are to be displayed in the preview window. FALSE indicates no controls are to be displayed.

Remarks

This method toggles the display of the preview window control buttons. Use this to display default controls, or to hide default controls and provide customized controls.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEShow Functions, Page 342

CRPEJob::StartJob

Syntax

```
StartJob () RETURNING BOOLEAN
```

Remarks

This method is used to start the processing of the print job and display the final output of that processing. This is the method that actually prints or exports the report.

Return Value

TRUE if the job is started successfully, FALSE if something goes wrong.

Related Topics

PEStartPrintJob, Page 348

CRPEJob::TestNthTableConnectivity

Syntax

```
TestNthTableConnectivity (tableN SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
tableN	Specifies the number of the table for which you want to test the connection settings. The first table added to a report is numbered 0, the second is 1, etc.

Remarks

This method tests to see if a valid connection exists to the database for the specified table in the report. This method is typically used if you plan to print at a later time, but you want to test now to make sure everything is in order for logging on to the database table.

Return Value

TRUE if the database session, log on, and location information is all correct. FALSE if something is wrong.

Related Topics

PETestNthTableConnectivity, Page 349

CRPEJob::ZoomPreviewWindow

Syntax

```
ZoomPreviewWindow (level SMALLINT) RETURNING BOOLEAN
```

Parameters

Parameter	Description
level	The magnification level to which the preview window is to be “zoomed”. Use one of the following values:
Value	Meaning
PEP_ZOOM_FULL_SIZE	Full page.
PEP_ZOOM_SIZE_FIT_ONE_SIDE	Fit one side (highest magnification).
PEP_ZOOM_SIZE_FIT_BOTH_SIDES	Fit both sides.

Remarks

This method “zooms” the preview window to the specified magnification level. Use this function when the report has been printed to a preview window and you need to set the magnification of the preview window to a specific level.

Return Value

TRUE if the call is successful, FALSE if something goes wrong.

Related Topics

PEZoomPreviewWindow, Page 351

CLASS CRPEExportOptions

This class is used to get and set the export options of a print job. It is used by member functions *CRPEJob::GetExportOptions*, [Page 976](#), and *CRPEJob::ExportTo*, [Page 974](#).

Data Members

<i>Member</i>	<i>Description</i>
<code>m_formatType</code>	Specifies the export format to be used.
<code>m_formatDLLName</code>	Specifies the name of the format DLL that contains the export format to be used.
<code>m_formatOptions</code>	Provides additional export information specific to the format type being used.
<code>m_nFormatOptionsBytes</code>	Automatically assigned by <code>CRPEJob::GetExportOptions</code> . Unused by <code>CRPEJob::SetExportOptions</code> .
<code>m_destinationType</code>	Specifies the destination type of the exported report.
<code>m_destinationDLLName</code>	Specifies the name of the destination DLL that contains the destination type to be used.
<code>m_destinationOptions</code>	Provides additional information specific to the export destination type.
<code>m_nDestinationOptionsBytes</code>	Automatically assigned by <code>CRPEJob::GetExportOptions</code> . Unused by <code>CRPEJob::SetExportOptions</code> .

Construction

<i>Constructor</i>	<i>Description</i>
<code>CRPEExportOptions</code>	Constructs a <code>CRPEExportOptions</code> class object.

CRPEExportOptions::CRPEExportOptions

Syntax

```
CRPEExportOptions ();
```

For example:

```
CRPEExportOptions (formatDLLName CHAR(*) : NULL, formatType INTEGER: 0,  
formatOptions FOREIGN : NULL, destinationDLLName CHAR(*) : NULL,  
destinationType INTEGER : 0, destinationOptions FOREIGN : NULL);
```

Parameters

<i>Parameter</i>	<i>Description</i>																																								
formatDLLName	Specifies the name of the format DLL that contains the export format to be used. Assigns this value to the CRPEExportOptions::m_formatDLLName member. Select a DLL from the following table:																																								
	<table border="1"> <thead> <tr> <th><i>To export a report in this format:</i></th><th><i>Use this DLL:</i></th></tr> </thead> <tbody> <tr> <td>Seagate Crystal Reports Format</td><td>uxfcr.dll</td></tr> <tr> <td>Data Interchange Format</td><td>uxfdif.dll</td></tr> <tr> <td>Word for Windows Format</td><td>uxfwordw.dll</td></tr> <tr> <td>Word for DOS Format</td><td>uxfdoc.dll</td></tr> <tr> <td>WordPerfect Format</td><td>uxfdoc.dll</td></tr> <tr> <td>Quattro Pro 5.0 (WB1) Format</td><td>uxfqp.dll</td></tr> <tr> <td>Record Style Format (columns)</td><td>uxfrec.dll</td></tr> <tr> <td>Rich Text Format</td><td>uxfrtf.dll</td></tr> <tr> <td>Comma Separated Values (CSV)</td><td>uxfsepv.dll</td></tr> <tr> <td>Tab Separated Values</td><td>uxfsepv.dll</td></tr> <tr> <td>Character Separated Values</td><td>uxfsepv.dll</td></tr> <tr> <td>Text Format (ASCII)</td><td>uxftext.dll</td></tr> <tr> <td>Tab Separated Text Format</td><td>uxftext.dll</td></tr> <tr> <td>Lotus 1-2-3 (WKS)</td><td>uxfwks.dll</td></tr> <tr> <td>Lotus 1-2-3 (WK1)</td><td>uxfwks.dll</td></tr> <tr> <td>Lotus 1-2-3 (WK3)</td><td>uxfwks.dll</td></tr> <tr> <td>Excel 2.1</td><td>uxfxls.dll</td></tr> <tr> <td>Excel 3.0</td><td>uxfxls.dll</td></tr> <tr> <td>Excel 4.0</td><td>uxfxls.dll</td></tr> </tbody> </table>	<i>To export a report in this format:</i>	<i>Use this DLL:</i>	Seagate Crystal Reports Format	uxfcr.dll	Data Interchange Format	uxfdif.dll	Word for Windows Format	uxfwordw.dll	Word for DOS Format	uxfdoc.dll	WordPerfect Format	uxfdoc.dll	Quattro Pro 5.0 (WB1) Format	uxfqp.dll	Record Style Format (columns)	uxfrec.dll	Rich Text Format	uxfrtf.dll	Comma Separated Values (CSV)	uxfsepv.dll	Tab Separated Values	uxfsepv.dll	Character Separated Values	uxfsepv.dll	Text Format (ASCII)	uxftext.dll	Tab Separated Text Format	uxftext.dll	Lotus 1-2-3 (WKS)	uxfwks.dll	Lotus 1-2-3 (WK1)	uxfwks.dll	Lotus 1-2-3 (WK3)	uxfwks.dll	Excel 2.1	uxfxls.dll	Excel 3.0	uxfxls.dll	Excel 4.0	uxfxls.dll
<i>To export a report in this format:</i>	<i>Use this DLL:</i>																																								
Seagate Crystal Reports Format	uxfcr.dll																																								
Data Interchange Format	uxfdif.dll																																								
Word for Windows Format	uxfwordw.dll																																								
Word for DOS Format	uxfdoc.dll																																								
WordPerfect Format	uxfdoc.dll																																								
Quattro Pro 5.0 (WB1) Format	uxfqp.dll																																								
Record Style Format (columns)	uxfrec.dll																																								
Rich Text Format	uxfrtf.dll																																								
Comma Separated Values (CSV)	uxfsepv.dll																																								
Tab Separated Values	uxfsepv.dll																																								
Character Separated Values	uxfsepv.dll																																								
Text Format (ASCII)	uxftext.dll																																								
Tab Separated Text Format	uxftext.dll																																								
Lotus 1-2-3 (WKS)	uxfwks.dll																																								
Lotus 1-2-3 (WK1)	uxfwks.dll																																								
Lotus 1-2-3 (WK3)	uxfwks.dll																																								
Excel 2.1	uxfxls.dll																																								
Excel 3.0	uxfxls.dll																																								
Excel 4.0	uxfxls.dll																																								
<i>Parameter</i>	<i>Description</i>																																								
formatType	Specifies the export format to be used. Assigns this value to the CRPEExportOptions::m_formatType member. Select from the following values:																																								

<i>To export a report in this format:</i>	<i>Use this for formatType:</i>
Seagate Crystal Reports Format	UXFCrystalReportType
Data Interchange Format	UXFDIFType
Word for Windows Format	UXFWordWinType
Word for DOS Format	UXFWordDosType
WordPerfect Format	UXFWordPerfectType
Quattro Pro 5.0 (WB1) Format	UXFQP5Type
Record Style Format (columns)	UXFRecordType
Rich Text Format	UXFRichTextFormatType
Comma Separated Values (CSV)	UXFCommaSeparatedType
Tab Separated Values	UXFTabSeparatedType
Character Separated Values	UXFCharSeparatedType
Text Format (ASCII)	UXFTextType
Tab Separated Text Format	UXFTabbedTextType
Lotus 1-2-3 (WKS)	UXFLotusWksType
Lotus 1-2-3 (WK1)	UXFLotusWk1Type
Lotus 1-2-3 (WK3)	UXFLotusWk3Type
Excel 2.1	UXFXls2Type
Excel 3.0	UXFXls3Type
Excel 4.0	UXFXls4Type

<i>Parameter</i>	<i>Description</i>
formatOptions	Provides additional export information specific to the format type being used. Assigns this value to the CRPEExportOptions::m_formatOptions member. This parameter is required for only certain format types. If you assign NULL to this parameter, the Crystal Report Engine will automatically prompt the user for format information if needed. Otherwise, use a format options class from the following table:

<i>To export a report in this format:</i>	<i>Use this class:</i>
Data Interchange Format	UXFDIFOptions

<i>To export a report in this format:</i>	<i>Use this class:</i>
Record Style Format (columns)	UXFRecordStyleOptions
Comma Separated Values (CSV)	UXFCommaTabSeparatedOptions
Tab Separated Values	UXFCommaTabSeparatedOptions
Character Separated Values	UXFCharSeparatedOptions

<i>Parameter</i>	<i>Description</i>
destinationDLLName	Specifies the name of the destination DLL that contains the destination type to be used. Assigns this value to the CRPEExportOptions::m_destinationDLLName member. The following options are available:

<i>To export a report to this destination:</i>	<i>Use this DLL name:</i>
Disk File	uxddisk.dll
E-mail (MAPI)	uxdmapi.dll
E-mail (VIM)	uxdvim.dll

<i>Parameter</i>	<i>Description</i>
destinationType	Specifies the destination type of the exported report. Assigns this value to the CRPEExportOptions::m_destinationType member. The following values are available:

<i>To export a report to this destination:</i>	<i>Use this destination type:</i>
Disk File	UXDDDiskType
E-mail (MAPI)	UXDMapiType
E-mail (VIM)	UXDVIMType

<i>Parameter</i>	<i>Description</i>
destinationOptions	Provides additional information specific to the export destination type. Assigns this value to the CRPEExportOptions::m_destinationOptions member. If you assign NULL to this parameter, the Crystal Report Engine will automatically prompt the user for destination information when needed. Otherwise, use a destination options class from the following table:

<i>To export a report to this destination:</i>	<i>Use this class:</i>
Disk File	UXDDiskOptions
E-mail (MAPI)	UXDMAPIOptions
E-mail (VIM)	UXDVIMOptions

Remarks

Constructs a CRPEExportOptions class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPEGraphDataInfo

The CRPEGraphDataInfo class contains information on what report data is used by a graph in the report to create the values in the graph. The class is used by *CRPEJob::GetGraphData*, *Page 977*, to retrieve information regarding an existing graph and by *CRPEJob::SetGraphData*, *Page 1016*, to change the data used by an existing graph.

Data Members

<i>Member</i>	<i>Description</i>
m_rowGroupN	Specifies which group number in the report is used to create the values in the rows of the graph.
m_colGroupN	Specifies which group number in the report is used to create the values in the columns of the graph.
m_summarizedFieldN	Specifies which summary field in the report is used to set the values of the risers in the graph. Summary fields are numbered in order of their creation.
m_graphDirection	Specifies the graphing direction for cross-tab reports. For normal group/total report, the direction, is always PE_GRAPH_COLS_ONLY.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEGraphDataInfo	Constructs a CRPEGraphDataInfo class object.

CRPEGraphDataInfo::CRPEGraphDataInfo

Syntax

```
CRPEGraphDataInfo (rowGroupN SMALLINT : 0, colGroupN SMALLINT: 0  
summarizedFieldN SMALLINT : 0, graphDirection SMALLINT :  
PEP_GRAPH_COLS_ONLY);
```

Parameters

Parameter	Description
rowGroupN	Specifies which group number in the report is used to create the values in the rows of the graph. Assigns this value to the CRPEGraphDataInfo::m_rowGroupN member variable.
colGroupN	Specifies which group number in the report is used to create the values in the columns of the graph. Assigns this value to the CRPEGraphDataInfo::m_colGroupN member variable.
summarizedFieldN	Specifies which summary field in the report is used to set the values of the risers in the graph. Summary fields are numbered in order of their creation. Assigns this value to the CRPEGraphDataInfo::m_summarizedFieldN member variable.
graphDirection	Specifies the graphing direction for cross-tab reports. For normal group/total report, the direction, is always PEP_GRAPH_COLS_ONLY = 1. Assigns this value to the CRPEGraphDataInfo::m_graphDirection member variable. Possible values are:

Constant	Value	Meaning
PEP_GRAPH_ROWS_ONLY	0	Use only row values in graph.
PEP_GRAPH_COLS_ONLY	1	Use only column values in graph.
PEP_GRAPH_MIXED_ROW_COL	2	Graph by row values, then by column values.
PEP_GRAPH_MIXED_COL_ROW	3	Graph by column values, then by row values.
PEP_GRAPH_UNKNOWN_DIRECTION	20	The direction of the graph is unknown.

Remarks

Constructs a CRPEGraphDataInfo class object. Call the constructor with no parameters to allow the

Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPEGraphOptions

The CRPEGraphOptions class contains information on several options available with graphs and charts. This class is used by *CRPEJob::GetGraphOptions*, [Page 978](#), to find out what options have been set for a graph, and it is used by *CRPEJob::SetGraphOptions*, [Page 1018](#), to change the options for a graph.

Data Members

<i>Member</i>	<i>Description</i>
m_graph.MaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted.
m_graph.MinValue	Specifies the minimum value that will appear in the graph. Any graph values below this value are not charted.
m_showDataValue	Specifies whether or not to display the numeric value associated with each riser on the chart. If set to TRUE (1), a value appears in the graph for each riser.
m_showGridLine	Specifies whether or not to display grid lines on the graph.
m_verticalBars	Specifies whether to display the bars in a bar graph vertically or horizontally.
m_showLegend	Specifies whether or not to display the graph legend.
m_fontFaceName	Specifies the font for all text and values in the entire graph.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEGraphOptions	Constructs a CRPEGraphOptions class type.

CRPEGraphOptions::CRPEGraphOptions

Syntax

```
CRPEGraphOptions()
```

For example:

```
CRPEGraphOptions (graph.MaxValue FLOAT : 0, graph.MinValue FLOAT : 0,
```

```

showDataValue BOOLEAN : FALSE, showGridLine BOOLEAN : FALSE,
verticalBars BOOLEAN : FALSE, showLegend BOOLEAN : FALSE, fontFaceName
CHAR (*) : NULL)

```

Parameters

Parameter	Description
graph.MaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted. Assigns this value to the CRPEGraphOptions::m_graph.MaxValue member variable.
graph.MinValue	Specifies the minimum value that will appear in the graph. Any graph values below this value are not charted. Assigns this value to the CRPEGraphOptions::m_graph.MinValue member variable.
show.MaxValue	Specifies whether or not to display the numeric value associated with each riser on the chart. If set to TRUE, a value appears in the graph for each riser. Assigns this value to the CRPEGraphOptions::m_show.MaxValue member variable.
show.GridLine	Specifies whether or not to display grid lines on the graph. Assigns this value to the CRPEGraphOptions::m_show.GridLine member variable.
vertical.Bars	Specifies whether to display the bars in a bar graph vertically or horizontally. Assigns this value to the CRPEGraphOptions::m_vertical.Bars member variable.
show.Legend	Specifies whether or not to display the graph legend. Assigns this value to the CRPEGraphOptions::m_show.Legend member variable.
font.FaceName	Specifies the font for all text and values in the entire graph. Assigns this value to the CRPEGraphOptions::m_font.FaceName member variable.

Remarks

Constructs a CRPEGraphOptions class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPEGraphTextInfo

The CRPEGraphTextInfo class contains information about the text that appears with a graph. This class is used with the methods *CRPEJob::GetGraphText*, *Page 980*, and *CRPEJob::SetGraphText*, *Page*

Data Members

<i>Member</i>	<i>Description</i>
m_graphTitle	Specifies the main title text that will appear above your graph.
m_graphSubTitle	Specifies the sub title text that will appear directly under the main title.
m_graphFootNote	Specifies the footnote text that will appear under your graph.
m_graphGroupsTitle	Specifies the title of the groups which are being graphed.
m_graphSeriesTitle	Specifies the title of the series which is being graphed.
m_graphXAxisTitle	Specifies the title text that will appear for the X axis. Not valid for Pie graphs.
m_graphYAxisTitle	Specifies the title text that will appear for the Y axis. Not valid for Pie graphs.
m_graphZAxisTitle	Specifies the title text that will appear for the Z axis. This value is only valid for 3D graphs.

Construction

<i>Constructor</i>	<i>Description</i>
CRPEGraphTextInfo	Constructs a CRPEGraphTextInfo class object.

CRPEGraphTextInfo::CRPEGraphTextInfo

Syntax

```
CRPEGraphTextInfo ();
```

For example:

```
CRPEGraphTextInfo (graphSubTitle CHAR(*) : NULL, graphFootNote CHAR(*) : NULL, graphGroupsTitle CHAR(*) : NULL, graphSeriesTitle CHAR(*) : NULL, graphXAxisTitle CHAR(*) : NULL, graphYAxisTitle CHAR(*) : NULL, graphZAxisTitle CHAR(*) : NULL);
```

Parameters

Parameter	Description
graphTitle	Specifies the main title text that will appear above your graph. Assigns this title to the CRPEGraphTextInfo::m_graphTitle member variable.
graphSubTitle	Specifies the sub title text that will appear directly under the main title. Assigns this title to the CRPEGraphTextInfo::m_graphSubTitle member variable.
graphFootNote	Specifies the footnote text that will appear under your graph. Assigns this text to the CRPEGraphTextInfo::m_graphFootNote member variable.
graphGroupsTitle	Specifies the title of the groups which are being graphed. Assigns this title to the CRPEGraphTextInfo::m_graphGroupsTitle member variable.
graphSeriesTitle	Specifies the title of the series which is being graphed. Assigns this title to the CRPEGraphTextInfo::m_graphSeriesTitle member variable.
graphXAxisTitle	Specifies the title text that will appear for the X axis. Not valid for Pie graphs. Assigns this title to the CRPEGraphTextInfo::m_graphXAxisTitle member variable.
graphYAxisTitle	Specifies the title text that will appear for the Y axis. Not valid for Pie graphs. Assigns this title to the CRPEGraphTextInfo::m_graphYAxisTitle member variable.
graphZAxisTitle	Specifies the title text that will appear for the Z axis. Only valid for 3D graphs. Assigns this title to the CRPEGraphTextInfo::m_graphZAxisTitle member variable.

Remarks

Constructs a CRPEGraphTextInfo class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPEJobInfo

This class is used by the member function *CRPEJob::GetJobStatus*, *Page 987*. Status information concerning the job is returned through this class.

Data Members

Member	Description
m_numRecordsRead	Specifies the number of records actually processed.
m_numRecordsSelected	Specifies the number of records selected for inclusion in the report out of the total number of records read.
m_numRecordsPrinted	Specifies the number of records actually printed.
m_displayPageN	Specifies the page number of the currently displayed page in the preview window.
m_latestPageN	Specifies the number of the last page that is currently available. Once the printing is complete, this value is the number of the last page.
m_startPageN	Specifies the number of the starting page. The value will normally be 1.
m_printEnded	Specifies whether or not the printing process is completed.

Construction

Constructor	Description
CRPEJobInfo	Constructs a CRPEJobInfo class object.

Upon construction of the structure, all members are initialized to zero except in printEnded which is initialized to FALSE.

CRPEJobInfo::CRPEJobInfo

Syntax

```
CRPEJobInfo ( numRecordsRead INTEGER : 0, numRecordsSelected INTEGER : 0, numRecordsPrinted INTEGER : 0, displayPageN SMALLINT : 0, latestPageN SMALLINT : 0, startPageN SMALLINT : 0, printEnded BOOLEAN : FALSE );
```

Remarks

Constructs a CRPEJobInfo class object. Initializes all required members of the class.

CLASS CRPELogOnInfo

This class is used by member functions *CRPEngine::LogOnServer*, *Page 964*, *CRPEngine::LogOffServer*,

Page 964, CRPEJob::GetNthTableLogOnInfo, Page 1000, CRPEJob::SetNthTableLogOnInfo, Page 1032. The class is used to contain SQL connection information.

Data Members

<i>Member</i>	<i>Description</i>
m_serverName	Specifies the logon name for the server used to create the report.
m_databaseName	Specifies the logon name for the database used to create the report.
m(userID)	Specifies the user ID necessary to log on to the server.
m_password	Specifies the password necessary to log on to the server.

Construction

<i>Constructor</i>	<i>Description</i>
CRPELogOnInfo	Constructs a CRPELogOnInfo class object.

CRPELogOnInfo::CRPELogOnInfo

Syntax

```
CRPELogOnInfo ();
```

For example:

```
CRPELogOnInfo (serverName CHAR (*), databaseName CHAR (*), userID CHAR (*), password CHAR (*));
```

Parameters

<i>Parameter</i>	<i>Description</i>
serverName	Specifies the logon name for the server or ODBC Datasource Name used to logon to the server. Assigns this value to the CRPELogOnInfo::m_serverName member.
databaseName	Specifies the logon name for the database used to run the report. Assigns this value to the CRPELogOnInfo::m_databaseName member.
userID	Specifies the user ID necessary to log on to the server. Assigns this value to the CRPELogOnInfo::m(userID) member.

<i>Parameter</i>	<i>Description</i>
password	Specifies the password necessary to log on to the server. Assigns this value to the CRPELogOnInfo::m_password member.

Remarks

Constructs a CRPELogOnInfo class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPEPrintOptions

This class is used to set the print characteristics of a report. It is used with *CRPEJob::SetPrintOptions*, *Page 1035*

Data Members

<i>Member</i>	<i>Description</i>
m_startPageN	Specifies the first page that you want to print.
m_stopPageN	Specifies the last page that you want to print.
m_nReportCopies	Specifies the number of copies that you want to print.
m_collation	Indicates whether or not you want the copies of the report to be collated (if you are printing multiple copies of a multiple page report).

Construction

<i>Constructor</i>	<i>Description</i>
CRPEPrintOptions	Constructs a CRPEPrintOptions class object.

CRPEPrintOptions::CRPEPrintOptions

Syntax

```
CRPEPrintOptions ();
```

For example:

```
CRPEPrintOptions (startPageN SMALLINT : 0, stopPageN SMALLINT : 0,
```

```
nReportCopies SMALLINT : 0, collation SMALLINT : PEP_DEFAULTCOLLATION);
```

Parameters

Parameter	Description
startPageN	Specifies the first page that you want to print. Assigns this value to the CRPEPrintOptions::m_startPageN member.
stopPageN	Specifies the last page that you want to print. Assigns this value to the CRPEPrintOptions::m_stopPageN member.
nReportCopies	Specifies the number of copies that you want to print. Assigns this value to the CRPEPrintOptions::m_nReportCopies member.
collation	Indicates whether or not you want the copies of the report to be collated (if you are printing multiple copies of a multiple page report). Assigns this value to the CRPEPrintOptions::m_collation member. Possible values are:

Value	Meaning
PEP_UNCOLLATED	Prints multiple copies of a multiple page report uncollated (Page order = 1, 1, 1, 2, 2, 2, 3, 3, 3, etc.).
PEP_COLLATED	Prints multiple copies of a multiple page report collated (Page order = 1, 2, 3..., 1, 2, 3..., etc.).
PEP_DEFAULTCOLLATION	Prints multiple copies of a multiple page report using the collation settings specified in the report.

Remarks

Constructs a CRPEPrintOptions class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPESectionOptions

This class contains specifications for formatting selected report sections. This information is used by the methods *CRPEJob::GetSectionFormat*, *Page 1003*, and *CRPEJob::SetSectionFormat*, *Page 1036*.

Data Members

Member	Description
m_visible	Specifies whether or not the selected section is to be visible. Use TRUE to keep the section visible, FALSE to hide the section.
m_newPageBefore	Specifies whether or not the Crystal Report Engine is to insert a page break before the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break.
m_newPageAfter	Specifies whether or not the Crystal Report Engine is to insert a page break after the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break.
m_keepTogether	Specifies whether or not the Crystal Report Engine is to keep all lines of the section together, either on the current page (if there is room), or on the next page. Pass TRUE to keep the lines together, FALSE to allow the Crystal Report Engine to split section data from one page to the next if necessary.
m_suppressBlankLines	Specifies whether or not the Crystal Report Engine is to eliminate lines from your report that are blank due to fields being suppressed (zeroes, duplicates, and hidden fields). TRUE eliminates blank lines. FALSE retains them.
m_resetPageNAfter	Specifies whether or not the Crystal Report Engine is to reset the page number to one (1) for the following page, after it prints a group total. Use TRUE to reset the page number, FALSE to keep standard numbering.
m_printAtBottomOfPage	Specifies whether or not the Crystal Report Engine is to cause each group summary value to print only at the bottom of a page. Use TRUE to print summaries at the bottom of the page, FALSE to print summaries immediately after the group.

Construction

Constructor	Description
CRPESectionOptions	Constructs a CRPESectionOptions class object.

CRPESectionOptions::CRPESectionOptions

Syntax

```
CRPESectionOptions (visible BOOLEAN : FALSE, newPageBefore BOOLEAN :  
FALSE, newPageAfter BOOLEAN : FALSE, keepTogether BOOLEAN : FALSE,
```

```

suppressBlankLines BOOLEAN : FALSE, resetPageNAfter BOOLEAN : FALSE,
printAtBottomOfPage BOOLEAN : FALSE)

```

Parameters

Parameter	Description
visible	Specifies whether or not the selected section is to be visible. Use TRUE to keep the section visible, FALSE to hide the section. Assigns this value to the CRPESectionOptions::m_visible member.
newPageBefore	Specifies whether or not the Crystal Report Engine is to insert a page break before the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break. Assigns this value to the CRPESectionOptions::m_newPageBefore member.
newPageAfter	Specifies whether or not the Crystal Report Engine is to insert a page break after the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break. Assigns this value to the CRPESectionOptions::m_newPageAfter member.
keepTogether	Specifies whether or not the Crystal Report Engine is to keep all lines of the section together, either on the current page (if there is room), or on the next page. Pass TRUE to keep the lines together, FALSE to allow the Crystal Report Engine to split section data from one page to the next if necessary. Assigns this value to the CRPESectionOptions::m_keepTogether member.
suppressBlankLines	Specifies whether or not the Crystal Report Engine is to eliminate lines from your report that are blank due to fields being suppressed (zeroes, duplicates, and hidden fields). TRUE eliminates blank lines. FALSE retains them. Assigns this value to the CRPESectionOptions::m_suppressBlankLines member.
resetPageNAfter	Specifies whether or not the Crystal Report Engine is to reset the page number to one (1) for the following page, after it prints a group total. Use TRUE to reset the page number, FALSE to keep standard numbering. Assigns this value to the CRPESectionOptions::m_resetPageNAfter member.
printAtBottomOfPage	Specifies whether or not the Crystal Report Engine is to cause each group summary value to print only at the bottom of a page. Use TRUE to print summaries at the bottom of the page, FALSE to print summaries immediately after the group. Assigns this value to the CRPESectionOptions::m_printAtBottomOfPage member.

Remarks

- Constructs a CRPESectionOptions class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.
- Use PEP_UNCHANGED with any parameter to let the Crystal Report Engine leave the setting as it is specified in the report file.

CLASS CRPESessionInfo

This class is used to get and set the session information (user ID and password) for password protected Microsoft Access databases. It is used with the member functions *CRPEJob::GetNthTableSessionInfo, Page 1000*, and *CRPEJob::SetNthTableSessionInfo, Page 1033*.

Data Members

<i>Member</i>	<i>Description</i>
m(userID)	Specifies the user ID needed for logging on to the MS Access system.
m(password)	Specifies the password needed for logging on to the MS Access system.
m(sessionHandle)	The handle to the current MS Access session.

Construction

<i>Constructor</i>	<i>Description</i>
CRPESessionInfo	Constructs a CRPESessionInfo class object.

CRPESessionInfo::CRPESessionInfo

Syntax

```
CRPESessionInfo ( );
```

For example:

```
CRPESessionInfo ( userID CHAR(*) : NULL, password CHAR(*) : NULL,  
sessionHandle INTEGER : 0 );
```

Parameters

Parameter	Description
userID	Specifies the user ID needed for logging on to the MS Access system. Assigns this value to the CRPESessionInfo::m(userID) member.
password	Specifies the password needed for logging on to the MS Access system. Assigns this value to the CRPESessionInfo::m(password) member.
sessionHandle	The handle to the current MS Access session. Assigns this value to the CRPESessionInfo::m(sessionHandle) member.

Remarks

Constructs a CRPESessionInfo class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPETableLocation

This class is used to get and set table location information. It is used with *CRPEJob::GetNthTableLocation*, *Page 999*, and *CRPEJob::SetNthTableLocation*, *Page 1032*.

Data Members

Member	Description
m_location	Specifies the database location.

Construction

Constructor	Description
CRPETableLocation	Constructs a CRPETableLocation class object.

CRPETableLocation::CRPETableLocation

Syntax

```
CRPETableLocation (location CHAR(*) : NULL);
```

Parameters

Parameter	Description
location	Specifies the database location. Assigns this value to the CRPETableLocation::m_location member.

Remarks

Constructs a CRPETableLocation class object. Call the constructor with no parameters to allow the Class Library to initialize all member variables with default values. Pass parameters to the constructor to assign specific values to each member variable.

CLASS CRPETableType

This class is used to determine information about a specific table used in the report. It is used by *CRPEJob::GetNthTableSessionInfo, Page 1000*.

Data Members

Member	Description
m_dllName	Specifies the name of the appropriate database DLL for the table of interest.
m_descriptiveName	Specifies the name of the table of interest.
m_dbType	Specifies the type of database that contains the table of interest.

Construction

Constructor	Description
CRPETableType	Constructs a CRPETableType class object.

CRPETableType::CRPETableType

Syntax

```
CRPETableType ( dbType SMALLINT : PEP_DT_SQL, dllName CHAR(*) : NULL,  
descriptiveName: CHAR(*) : NULL);
```

Remarks

Constructs a CRPETableType class object. Initializes all members of the class.

Part 4 - User Defined Functions

13

Creating User Defined Functions

What you will find in this chapter...

Designing User Defined Functions, Page 1066
Programming the UFL, Page 1069
Helper Modules, Page 1069
Setting Up a UFL Project, Page 1070
Function Definition, Page 1071
Function Definition Table, Page 1072
Function Templates Table, Page 1073
Function Examples Table, Page 1074
Error Table, Page 1075
InitForJob Function, Page 1076
TermForJob Function, Page 1076
UFL Function Implementation, Page 1077
Returning User Defined Errors, Page 1077
Obtaining parameter values from the parameter block, Page 1078
Picture Function - a sample UFL function, Page 1079
Module Definition (.def) File, Page 1082
UFJOB Modules, Page 1083
Known problems with earlier versions of UFJob.C, Page 1085

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

The Seagate Crystal Reports Formula Editor and formula language are powerful tools, enabling you to perform a wide variety of report-related tasks easily and efficiently. The formula language is expandable as well. That is, while it already includes a large selection of useful functions, it also comes with the ability to accept new functions that you define to meet your needs.

User Defined Functions that are recognized by the Seagate Crystal Reports Formula Editor can be created in a Dynamic Link Library or, for 32-bit environments, in an Automation Server. This section demonstrates how to create User Defined functions in a Dynamic Link Library using the C programming language. For information on how to create User Defined Functions in an Automation Server using Visual Basic, see *Creating User Defined Functions in Visual Basic, Page 1088*.

Designing User Defined Functions

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

You can add new functions to the Seagate Crystal Reports Formula Editor by:

- writing the functions using the C programming language, and
- compiling and linking the functions into a User Defined Function DLL called a UFL (User Function Library).

NOTE: *If you are not familiar with programming Windows DLLs, refer to the Microsoft Windows Software Development Kit. Do not attempt to create a UFL if you do not understand Windows DLL programming.*

When designing a new function for the Seagate Crystal Reports Formula Editor, you need to determine the following:

- the name of the function,
- the purpose of the function,
- the type of data (report data or user supplied data) the function will require, and
- the type of data the function will return to the report.

Name of the function

The name of your new function should reflect the purpose of the function, making it easier to recognize when it appears in the Formula Editor's Functions list. For example, a function named "Picture" could let you specify a template picture of how data should appear in the report. If a field contains phone numbers, you can use the Picture function to specify that the data appear like this:

(xxx) xxx-xxxx

A resulting value from the phone number field would appear as follows:

(415) 555-1234

Function names must start with a letter, while all remaining characters in the name can be letters or numbers. The name can be up to 254 characters long, and it must be unique. That is, you can not give a function a name that has been used for another Formula Editor function or UFL function, nor can it have a name that matches a standard C keyword (such as **if**, **return**, **switch**, or **while**).

Purpose of the function

You may find it helpful to start simply by “fleshing out” your function. Determine the purpose of the function and outline it on paper. Use C code or even pseudocode to determine how the function will perform the required operation. This step is important, as it will form the base information for every other step in the designing and programming of your UFL.

Function data

UFL functions are much like any other function you might create in C. They can accept values that are passed as parameters, and they return a value that is printed on the report. Once you have determined how a UFL function will perform a task, you will know exactly what kind of data it will require to complete that operation. The following table shows every data type that a UFL function can accept as a parameter, along with a description of what the parameter will look like in C:

Parameter Type	C Data Type
number	Double.
currency	Double.
Boolean	Short integer.
string	Pointer to an array of characters.
range (number)	Structure containing two doubles.
range (currency)	Structure containing two doubles.
range (Boolean)	Structure containing two short integers.

<i>Parameter Type</i>	<i>C Data Type</i>
range (date)	Structure containing two long integers.
range (string)	Structure containing pointers to two elements in a character array.
array (number)	Pointer to a number array.
array (currency)	Pointer to a number array.
array (integer)	Pointer to an integer array.
array (Boolean)	Pointer to a Boolean array.
array (date)	Pointer to a date array.
array (string)	Pointer to a string array.

Return types

Finally, you must determine what kind of data your function returns to the current report in Seagate Crystal Reports. The following table lists possible UFL return types along with a description of the C data type used when programming the function.:

<i>Return Type</i>	<i>C Data Type</i>
string	Pointer to a character array.
number	Double.
date	Long integer.
Boolean	Short integer.
currency	Double.
range (date)	Structure containing two long integers.
range (number)	Structure containing two doubles.
range (currency)	Structure containing two doubles.
range (string)	Structure containing pointers to two elements in a character array.
array (date)	Pointer to a date array.
array (number)	Pointer to a number array.
array (currency)	Pointer to a number array.
array (string)	Pointer to a string array.
array (Boolean)	Pointer to a Boolean array.

Programming the UFL

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

After sketching out your new function, deciding on its parameters and data types, and determining the type of data the function will return to a report, you can begin programming the UFL.

A UFL is like any other DLL with a few simple differences:

- The file name must have the “UFL” prefix for 16-bit Seagate Crystal Reports, or with a “U2L” prefix for 32-bit Seagate Crystal Reports (i.e., UFLSAMP.DLL or U2LSAMP.DLL),
- it must export your User Defined Function (UDF) as a DLL function, and
- it must export a collection of other functions required by Seagate Crystal Reports.

You can design your UFL from scratch, but you may find the helper modules provided with Seagate Crystal Reports useful. If you use the helper modules provided, you will only need to create a single C code module, a module definition (.def) file, and an application project file. More experienced programmers may want to use these modules simply as a starting point to design more complex UFL features.

NOTE: *Although 32-bit versions of User Function Libraries have a “U2L” prefix, they are still referred to as UFLs.*

Helper Modules

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

The following files have been installed on your system in the same directory as CRW.EXE (C:\CRW by default):

<i>File Name</i>	<i>Purpose</i>
ufdll.h	Defines UFL enumerated, union, structure, and other data types.
ufuser.h	Prototypes of tables and functions user must implement.
ufmain.h	Prototypes of internal UFL functions implemented in ufmain.c.
uffuncs.h	Prototypes of functions that must be exported by the UFL to be used by Seagate Crystal Reports. These functions are implemented in UFMAIN.C.

<i>File Name</i>	<i>Purpose</i>
ufmain.c	Implementation of UFL functions used internally and used by Seagate Crystal Reports when connecting to the UFL. This file also contains generic LibMain and WEP functions for Windows 3.x DLLs and a generic DllEntryPoint function for Win32 DLLs. (The LibMain, WEP, and DllEntryPoint functions are defined conditionally according to whether or not you are building a Win32 DLL. You do not need to make any changes to the code here.)
ufjob.h	Optional file. Contains a definition of the JobInfo structure see <i>UFJOB Modules, Page 1083</i> and prototypes of the <i>InitForJob Function, Page 1076</i> , <i>TermForJob Function, Page 1076</i> , and <i>FindJobInfo</i> (see <i>Implementing InitJob and TermJob, Page 1085</i> implemented in UFJOB.C. structure and PROTOTYPES.C.
ufjob.c	Optional file. Contains implementations of the required <i>InitForJob Function</i> and <i>TermForJob Function</i> . Also implements the <i>FindJobInfo</i> helper function.

You do not need to work with the actual code in any of these files, but they will have to be added to your UFL project. UFJOB.H and UFJOB.C are optional files providing job information for the current print job accessing the Formula Editor. For more information, see *UFJOB Modules, Page 1083*.

Setting Up a UFL Project

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

Begin creating your UFL by setting up a new directory on your system to work in. Use File Manager or Windows Explorer to copy all of the files listed in the chart under *Helper Modules, Page 1069*, into your new directory. This assures that you do not inadvertently edit or change the original files in your CRW directory.

NOTE: *If you will not be using the functionality provided by UFJOB.H and UFJOB.C, you do not need to copy these files into your working directory.*

Now, open your Integrated Development Environment (IDE) application, and use the tools there to create a new project file. Name the new project file with a UFL prefix if your function library will be used with Seagate Crystal Reports for Windows 3.x, or with a U2L prefix if it will be used with Seagate Crystal Reports for Win32. For example, UFLSAMP.MAK or U2LFUNCS.PRJ. Make sure the project is set to build a DLL (not a Windows EXE) file.

IMPORTANT: If you are building a 32-bit version of a User Function Library, make sure you set Struct Member Alignment for the project to 1 byte. Seagate Crystal Reports for Win32 will not be able to use your UFL otherwise. To find out how to change the Struct Member Alignment setting for your project, refer to the documentation for your development environment.

Finally, add the UFMAIN.C file to your project, and save the project in your new working directory. Add the UFJOB.C file as well if you are using this file. You are now ready to begin creating a new UFL for the Seagate Crystal Reports Formula Editor.

Function Definition

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Creating a UFL function requires that you create only one more C code module and a module definition (.def) file in addition to the Helper modules. (Your particular UFL Function needs may require more modules, but the simplest method for creating a UFL requires creating only these two.) For information on creating the module definition file, see *Module Definition (.def) File, Page 1082*. To begin building your UFL, create a new C module in your IDE, and save it to your working directory with the same name as your project file. For example, if your project file is named UFLSAMP.MAK or UFLFUNCS.PRJ, you would name your C module UFLSAMP.C or UFLFUNCS.C respectively. This demonstration will refer to UFLSAMP.C. This is the “private” C module for your UFL because it is the one section of code that must be unique to your UFL.

The first step to programming your UFL’s private C code module is to #include the appropriate header files:

- #include <windows.h>
- #include “ufdll.h”
- #include “ufmain.h”
- #include “ufuser.h”

You do not need to #include the UFFUNCS.H header file that you also copied into your working directory. This file is already #included by UFMAIN.C, and you will not be directly calling any of the functions defined in these files (though they are necessary for Seagate Crystal Reports when the Formula Editor accesses your UFL).

The private C code module of your UFL requires several parts:

- *Function Definition Table, Page 1072,*
- *Function Templates Table, Page 1073,*
- *Function Examples Table, Page 1074,*
- *Error Table, Page 1075,*
- *InitForJob Function, Page 1076,*

- *TermForJob Function, Page 1076*, and
- *UFL Function Implementation, Page 1077*.

Most of these sections have specific guidelines that must be used and that are the same for every function you add to your UFL. Your UFL function implementation is completely designed and programmed by you. It is the functionality that you are adding to the Seagate Crystal Reports Formula Editor.

The “Picture” example

For the demonstration in this Help file of how to program a UFL, you will create the Picture function. The purpose of this function is to display string type field data using a format specified by the user. For example, if a phone number is entered in the database table as “4155551234”, the user can create the following formula:

```
Picture({table.PHONENUM}, "(xxx) xxx-xxxx")
```

The phone number will appear in the report as:

```
(415) 555-1234
```

Function Definition Table

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

You must supply a definition for each function that you want to add to Seagate Crystal Reports. Each entry in the function definition table consists of a definition string that specifies: the return type,

- the function name,
- an argument list (showing the required order in which arguments are to be entered and the data type of each argument), and
- the name of the C/C++ function that implements the call. The definition must appear with the following format:

```
"returnType UDFName (arg1, arg2,...)", CFunctionName
```

Here is an example:

```
"String Picture (String, String)", Picture
```

In this example:

- “String” specifies that the function is to return a string.
- “Picture” is the name that will identify the function on the Function list in the Formula Editor.

- “(String, String)” specifies that the function is to require two arguments, both are strings.
- “Picture” (appearing after the comma), is the actual function name, the name you give the function when you code it.

NOTE: *The UDFName and the CFunctionName do not have to be the same. You can use something other than the function name to identify a function on the Function list of the Formula Editor if you wish.*

All function definitions must be set up in a table with the following heading:

```
UFFunctionDefStrings FunctionDefStrings [ ] =
```

NOTE: *The table must be terminated with three nulls.*

Seagate Crystal Reports uses the information you supply in this table to create parameter blocks when you call the functions.

Function definition table example

Here is a sample function definition table for the Picture function:

```
UFFunctionDefStrings FunctionDefStrings [ ] =
{
  {"String Picture(String, String)", Picture},
  {NULL, NULL, NULL}
};
```

An optional third parameter can be used when you know the maximum length of the return value (or your code can obtain it). For example, the definition for the Picture function might be:

```
{"String Picture (String, String)", Picture, PictureRetSize},
```

Here, PictureRetSize specifies the maximum acceptable length of the string returned by the Picture function.

Function Templates Table



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

You must supply a function template for each function that you define. A function template specifies the string that Seagate Crystal Reports is to enter in the Formula Editor's Formula text box when you select the function from the Functions list in the Formula Editor. Each template must contain:

- the function call,
- any syntax guides (parentheses, commas, etc.) you want to include, and
- an exclamation point (!) character to specify where the insertion point is to appear when the function is entered into the formula. Here is an example:

```
"Picture (!, )"
```

In this example, the string “Picture (,)” is to be entered into the formula whenever you select the Picture function from the Functions list in the Formula Editor.

- The string includes the function call “Picture” and the syntax guides “(,)” to guide the user when entering arguments.
- The exclamation point specifies that the insertion point is to be placed inside the parentheses, before the comma.

All function templates must be set up in a table with the following heading:

```
UFFunctionTemplates FunctionTemplates [ ] =
```

NOTE: *The table must be terminated with a null.*

Continuing with the example, the function templates table for the Picture function should look like this:

```
UFFunctionTemplates FunctionTemplates [ ] =
{
    {"Picture (!, )"} ,
    {NULL}
};
```

Function Examples Table

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

You must also supply a function example for each function. A function example specifies the string you want to use to identify and select the function in the Functions list in the Formula Editor. Here is a function example for the Picture function:

```
"\tPicture (string, picture)"
```

This example specifies that the string “Picture (string, picture)” is the listing that you want to appear in the Functions list. The characters “\t” specify that the string is to be set in from the left one tab stop, thus aligning it with other functions in the list.

All function examples must be set up in a table with the following heading:

```
UFFunctionExamples FunctionExamples [] =
```

NOTE: *The table must be terminated with a null.*

The complete function examples table for the Picture function should look like this:

```
UFFunctionExamples FunctionExamples [] =
{
    {"\tPicture (string,picture)"},  
    {NULL}
};
```

Error Table

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

You must also supply an error string for each error message that you want to make available to your functions. Each error string contains only the text you want to display when an error is detected, and it must be in the format:

```
"ErrorString"
```

NOTE: *C compilers view the first string in the table as Error 0, the second as Error 1, etc. Each string in the error table corresponds to an error code that you define. For each user error code, there must be a message string in the error table at the corresponding index, where the first string is at index 0.*

Each UFL C function must return a value of the enumerated type UFError, defined in ufdll.h. Return UF.NoError if no error occurred. Return one of the other UFError values if there is an error. Try to choose one of the predefined values if it fits your situation. If the error is specific to your UFL, set the ReturnValue.UF.ReturnUserError member of the parameter block to an error code value you have defined and return UF.UserError from your function code. The Formula Editor will then call back to return the error string that you have defined in the error table.

The Formula Editor passes a parameter block to a UFL function rather than individual parameters. *Obtaining parameter values from the parameter block, Page 1078*, will examine how to handle parameter blocks.

All error strings must be set up in a table with the following heading:

```
char *ErrorTable [] =
```

An error table for the Picture function should, at a minimum, look like this:

```
char *ErrorTable[] =
{
    "no error"
};
```

InitForJob Function

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

The InitForJob function initializes all user function UFL's with the job ID whenever a job starts. You can handle any job initialization for your functions here, but all that is absolutely necessary is an empty function implementation:

```
void InitForJob(UFTInt32u jobID)
{
}
```

NOTE: See the note following TermForJob Function, Page 1076.

TermForJob Function

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

The TermForJob function terminates the job ID for all user function UFL's whenever a job finishes. You can add any job termination code here that you like, but all that is absolutely necessary is an empty function implementation:

```
void TermForJob(UFTInt32u jobID)
{
}
```

NOTE: Every UFL must have an implementation of InitForJob Function, Page 1076, and TermForJob. These functions are called when a job starts and ends printing, respectively. You can choose to implement these yourself. At a minimum, you must provide empty functions.

Seagate Crystal Reports provides helper modules (UFJOB.C and UFJOB.H) which implement these functions and maintain a doubly linked list of JobInfo structures, one for each active job. The JobInfo structure (declared in UFJOB.H) holds the ID# of the job and contains a void pointer where you can store any data that you allocate. The files also implement a FindJobInfo function (see *Implementing InitJob and TermJob, Page 1085*, which you can use to retrieve the job information for any open job). *UFJOB Modules, Page 1083*, will examine these files and their implementation of InitForJob and TermForJob.

UFL Function Implementation

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

As the final step to creating a UFL function (see *The Function Definition, Page 1071*), you must add code for the operation of the function you have designed. Your function must be programmed for the specific needs of your UFL. This section will examine the basics of how to obtain the parameters from the parameter block and use the values of those parameters in the implementation of the UFL function.

You begin by coding your function as follows:

```
ExternC UFError FAR _export PASCAL FunctionName  
    (UFParamBlock *ParamBlock)  
{  
    // Your function's code  
}
```

First, notice that the function is exported because a UFL is simply a DLL being accessed by Seagate Crystal Reports. Second, the function returns an error code of the enumerated type UFError. In the error trapping sections of your function, you can return a UFError value, such as UFNotEnoughParameters, or you can return UFUserError and define your own errors in the error table. Finally, the function accepts a parameter block of type UFParamBlock from Seagate Crystal Reports rather than individual parameters. You will need to retrieve the individual parameters from that parameter block to work with the data values passed by the Formula Editor.

NOTE: ExternC is defined in UFDLL.H and is equivalent to: extern "C".

Returning User Defined Errors

 *This topic is only supported by the Professional Edition of Seagate Crystal Reports.*

The UFError enumerated type provides several errors that are common to many types of functions. If appropriate, have your UFL function return one of these predefined types. If no error occurs, you can return UF.NoError.

If your function can cause an error that is not predefined, however, you can establish a user-defined error. You do this by:

- adding an error string to the error table,
- passing the appropriate error index to the ReturnValue.UFReturnUserError member of the parameter block, and

- returning UFUserError from your UFL function implementation.

A user-defined error string in the error table might look like this:

```
char *ErrorTable[] =
{
    "My User-defined Error"
};
```

This error string is assigned an error index by Seagate Crystal Reports. The first error is 0, the second is 1, etc. If an error occurs in your function, you assign the appropriate index value to the ReturnValue.UFReturnUserError member of the parameter block. For example:

```
ParamBlock->ReturnValue.UFReturnUserError = 0;
```

Once you specify a user-defined error, you can return the UFError value UFUserError from your function. When the Formula Editor finds the error in a formula entered in the Formula text box (when the Check or Accept button is clicked), it will use the error index specified to report the appropriate string listed in the error table.

Obtaining parameter values from the parameter block

 This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Since Seagate Crystal Reports passes the values for a Formula Editor function's parameters in a parameter block rather than individually, you must separate the values from the parameter block before they can be evaluated. For the Picture function (see *Picture Function - a sample UFL function, Page 1079*), expect the parameter block to contain two parameters, both of type String, as defined in the function definition table.

To obtain the values of the individual parameters, begin by defining pointers to two structures of type UFParamListElement (defined in UFDLL.H):

```
UFParamListElement*FirstParam,
*SecondParam;
```

Next, call the GetParam function (defined in UFMAIN.C) for each UFParamListElement to obtain a pointer to each parameter from the parameter block:

```
FirstParam = GetParam (ParamBlock, 1);
SecondParam = GetParam (ParamBlock, 2);
```

The actual value is stored in the Parameter member of the UFParamListElement according to the type of data it contains. The Parameter member is a UFParamUnion type (a union data type holding a value according to the type of data expected). Since both of the parameters are of type String, you can obtain the actual parameter value for each UFParamListElement by using the following notation:

```
FirstParam->Parameter.ParamString  
SecondParam->Parameter.ParamString
```

If, on the other hand, the second parameter contained a numeric value, you could use this notation:

```
SecondParam->Parameter.ParamNumber
```

Study the UFParamUnion union definition in the UFDLL.H file for a complete list of all possible parameter types and how to obtain a value from them.

Picture Function - a sample UFL function



This topic is only supported by the Professional Edition of Seagate Crystal Reports.

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Here is a complete commented private C code module implementing the Picture function. Use this as a guide for how to create your own functions in UFLs:

```
*****  
** UFLSAMP.C  
**  
** Private C code module  
** implementing the Picture UDF.  
*****  
  
#include <Windows.h>  
#include "UFDll.h"  
#include "UFMain.h"  
#include "UFUser.h"  
  
#define PicturePlaceholder 'x'  
  
/* UDF PROTOTYPE */  
ExternC UFError FAR _export PASCAL Picture  
(UFParamBlock * ParamBlock);  
  
*****  
* This array gives the program the types for the  
* parameters to the UDF, the return
```

```

* type, and the name. It also passes the
* address of the actual function code.
*****/

UFFFunctionDefStrings FunctionDefStrings [ ] =
{
    {"String Picture (String, String)",
     Picture},
    {NULL, NULL, NULL}
};

*****  

* The following is the template the program
* will insert into the Formula Editor
* Formula text box when this function is
* selected.
*****/  

UFFFunctionTemplates FunctionTemplates [ ] =
{
    {"Picture (!, "},
    {NULL}
};

*****  

* The following is an example of the format
* for this function. This text will appear
* in the Functions list box of the Formula
* Editor.
*****/  

UFFFunctionExamples FunctionExamples [ ] =
{
    {"\tPicture (string, picture)" },
    {NULL}
};

*****  

* This array contains ASCII string
* representations of the errors which
* could occur.
*****/  

char *ErrorTable [ ] =
{

```

```

        "no error"
    };

/* Called for on initialization */
void InitForJob (UFTInt32u jobID)
{
}

/* Called on termination */
void TermForJob (UFTInt32u jobID)
{
}

*****
* This function is used by the Picture UDF to
* copy the contents of a source string and a
* format string into a destination string.
*****/

static void copyUsingPicture(char *dest,
const char *source, const char *picture)
{
    while (*picture != '\0')
    {
        If (tolower (*picture) ==
PicturePlaceholder)
            If (*source != '\0')
                *dest++ = *source++;
            Else
                ; // do not insert anything
            Else
                *dest++ = *picture;
                picture++;
    }
    // copy the rest of the source
    lstrcpy (dest, source);
}

*****
*      This is the User Defined Function
*****/


ExternC UFError FAR _export PASCAL Picture
(UFParamBlock * ParamBlock)
{
    UFParamListElement*FirstParam,*SecondParam;
}

```

```

FirstParam = GetParam (ParamBlock, 1);
SecondParam = GetParam (ParamBlock, 2);

If (FirstParam == NULL || SecondParam ==
NULL)
    return UFNotEnoughParameters;
copyUsingPicture(ParamBlock-
ReturnValue.ReturnString,
FirstParam-Parameter.Parameter.String,
SecondParam-Parameter.Parameter.String);
return UFNoError;
}

```

Module Definition (.def) File

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

The last element of your UFL is the module definition (.def) file. This is just like any module definition file you would create for a DLL, but you must make sure to explicitly export not only your UFL function, but also the specialized UFL functions defined in UFMAIN.C that Seagate Crystal Reports expects to find. The following is an example of a module definition file for the UFL that exports the Picture function (see *Picture Function - a sample UFL function, Page 1079*):

```

Library      UFLSAMP
Description'User Function Library for Crystal Reports'
ExeType      Windows
HeapSize     1024
Code          Moveable Discardable Preload
Data          Moveable Preload
Segments     _TEXT  Preload

Exports
        WEP
        UFINITIALIZE
        UFTERMINATE
        UFGETVERSION
        UFSTARTJOB
        UFENDJOB
        UFGETFUNCTIONDEFSTRINGS
        UFGETFUNCTIONEXAMPLES
        UFGETFUNCTIONTEMPLATES
        UFERRORRECOVERY
        PICTURE

```

Notice that the only function exported that you actually coded is the Picture UFL. The rest of the exported functions have been defined for you in UFMAIN.C. Every UFL must export these 9 UF* functions. In addition to these functions, every UFL that you create must be exported.

NOTE: If you are creating a 32-bit UFL, do not export the WEP procedure function. WEP is specific to 16-bit Windows. If you are creating a 16-bit UFL, however, the WEP function must be exported here.

NOTE: If you are creating a 32-bit UFL, you will only need to include the LIBRARY, DESCRIPTION, and EXPORTS sections of the .def file. Any other sections included will be ignored by 32-bit Windows.

When you have finished coding the module definition file, save it to your working directory and add it to the list of files in your project file.

Finally, compile and link the ufl* or u2l* project file. Resolve any errors that occur, and recompile if necessary. Once you have your DLL (ufl*.dll or u2l*.dll), place it in the directory that holds CRW.EXE. From that point on, when you open the Formula Editor, your user defined function(s) will appear in the “Additional Functions” section at the bottom of the Functions list of the Formula Editor. Enter each function in one or more formulas, and test and modify it until it works the way you want.

NOTE: For additional information, review UFLSAMP1.C and UFLSAMP2.C (sample files that were installed in the Seagate Crystal Reports directory, \CRW, by default).

UFJOB Modules

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

Two optional modules, UFJOB.H and UFJOB.C have been provided with Seagate Crystal Reports. These files provide an implementation of the *InitForJob Function, Page 1076*, and *TermForJob Function, Page 1076*, which allow you to obtain an ID number that is specific to the current print job in Seagate Crystal Reports. At the same time, these modules establish a JobInfo structure for the current job where you can store information regarding the job. If your UFL, for example, must evaluate all values in a field before printing a result, it can tally data in the JobInfo structure until it has a result. Data can even be passed between functions using the JobInfo structure.

Use the JobInfo structure whenever you want to create UFL functions that summarize or group report data. For example, statistical functions that evaluate the median, mean, or range of values in a field can store data in the JobInfo structure.

The UFLSAMP2.C file, included with Seagate Crystal Reports, demonstrate UFLs that group data according to the Top N values. (Seagate Crystal Reports can do this automatically for you, but the functions in UFLSAMP2.C will help you understand how to use the functions and the JobInfo structure in the UFJOB modules.)

If you decide to use the UFJOB modules in your own UFL, you must do the following:

- add UFJOB.C to your UFL project file,
- #include UFJOB.H in your private C code module that implements your UFL functions, and
- replace your own implementations of the InitForJob and TermForJob functions with implementations of the InitJob and TermJob functions.

UFJOB.C

This module contains implementations of the *InitForJob Function*, *Page 1076*, and *TermForJob Function*, *Page 1076*, that provide a JobInfo structure for the current job in Seagate Crystal Reports. These replace any versions of these functions that you coded in your private C module containing your UFL definitions. They also call the InitJob and TermJob functions respectively. You will implement these functions in your private C code module.

In addition, this file also defines the FindJobInfo function. Use this function in your own code whenever you need to obtain a pointer to the JobInfo structure for the current job. This function requires only the job ID number, which is stored in the JobId member of the parameter block. The following code demonstrates how to call this function:

```
struct JobInfo *jobInfo;  
jobInfo = FindJobInfo (ParamBlock->JobId);
```

UFJOB.H

This module prototypes the *InitForJob Function*, *Page 1076*, *TermForJob Function*, *Page 1076*, and *FindJobInfo* functions that appear in UFJOB.C. It also prototypes the InitJob and TermJob functions that you must implement yourself. Most importantly, this file defines the JobInfo structure:

```
struct JobInfo  
{  
    UFTInt32u jobId;  
    struct JobInfo FAR *prev;  
    struct JobInfo FAR *next;  
  
    void FAR *data;  
};
```

The data member of this structure allows you to store a pointer to any kind of data you wish. This means you can store information for the current job that allows you to evaluate several field values before printing a result or store data from one function to be used by another function.

Implementing InitJob and TermJob

If you previously implemented the InitForJob and TermForJob functions in the private C code module with your UFL function definitions, delete those functions now since they are being replaced by the functions in UFJOB.C. Now, in your private C module, define the InitJob and TermJob functions which InitForJob and TermForJob call. You can use these functions to add job initialization or job termination code to your UFL, but they can also remain blank.

Following are examples of how you might implement these functions for your own UFL:

```
void InitJob (struct JobInfo *jobInfo)
{
}

void TermJob (struct JobInfo *jobInfo)
{
    If (jobInfo->data != 0)
        free (jobInfo->data);
}
```

Once you have done all this, you can make full use of the JobInfo structure with your own UFL function code.

Known problems with earlier versions of UFJob.C

Early versions of UFJob.C, a file used for creating UFL functions, can cause a UFL to go into an infinite loop under certain conditions, forcing a user to reboot his or her system. Problems can arise when using 16-bit UFL functions to handle multiple client reports simultaneously. The problem was fixed several years ago but it may recur with the current version of Seagate Crystal Reports if you are still using an early version of UFJob.C.

If you are experiencing the problem, you can correct it in one of two ways:

- Open your UFL project in your C or C++ development environment. Replace the UFJob.C module in your project with the current UFJob.C module distributed with Seagate Crystal Reports. (A comment appears at the beginning of the module code showing a date of 13 Mar 93.) Finally, recompile your UFL project.

- Open your UFL project in your C/C++ development environment and edit the code in UFJob.C. Replace the following code:

```
while (cursor !=0)
    if (cursor->jobId==jobId)
        break;
```

with this code

```
while (cursor !=0)
{
    if (cursor->jobId==jobId)
        break;
    cursor = cursor->next;
}
```

14

Creating User Defined Functions in Visual Basic

What you will find in this chapter...

Using Visual Basic 4.0, Page 1089

Using Visual Basic 5.0, Page 1091

Using the User Defined Functions, Page 1092

Visual Basic and Seagate Crystal Reports Variable Types, Page 1093

Using Arrays, Page 1093

Registering the Automation Server, Page 1092

Special Purpose Functions, Page 1094

Error Handling, Page 1093

Distribution, Page 1095

Sample UFL Automation Server, Page 1095

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

The Seagate Crystal Reports Formula Editor and formula language are powerful tools, enabling you to do a wide variety of report-related tasks easily and efficiently. The formula language is expandable as well. That is, while it already includes a large selection of useful functions, it also comes with the ability to accept new functions that you define to meet your needs.

User Defined Functions that are recognized by the Seagate Crystal Reports Formula Editor can be created in a Dynamic Link Library or, for 32-bit environments, in an Automation Server. This section demonstrates how to create User Defined Functions in an Automation Server using Visual Basic. For information on how to create User Defined Functions in a Dynamic Link Library using the C or C++ programming language, see *Creating User Defined Functions, Page 1065*. For information on creating User Defined Functions in an Automation Server using Delphi 3.0, see *Creating User Defined Functions in Delphi 3.0, Page 1099*.

Creating User Defined Functions in Visual Basic

The Seagate Crystal Reports Formula Editor can access User Defined Functions through a User Function Library (UFL). A User Function Library is a specially designed Dynamic Link Library that exposes one or more functions that you create. UFLs can be designed and programmed in any language that supports the development of Windows DLLs.

The 32-bit version of Seagate Crystal Reports Professional Edition includes the User Function Library U2LCOM.DLL. This UFL is installed in your \WINDOWS\CRYSTAL directory when you install Seagate Crystal Reports and provides an interface through which you can expose User Defined Functions in Automation Servers.

An automation server is a Dynamic Link Library or executable file that exposes its functionality to other modules and processes through the Component Object Model. For complete information on the Component Object Model (COM), refer to Microsoft documentation or the Microsoft World Wide Web site. U2LCOM.DLL is a UFL that can access any functions exposed by any Automation Servers named with a CRUFL prefix. This means that you can create an Automation Server in any language environment that supports COM, name the server with a CRUFL prefix, register the server on a system, and the Seagate Crystal Reports Formula Editor will access and make available any functions exposed by that Automation Server.

NOTE: You may be more familiar with Automation Servers as OLE Automation. Though the technology behind Automation Servers is the same as OLE Automation, the correct name for these servers is now simply Automation Servers.

Visual Basic, version 4.0 and later, allows you to design 32-bit Automation Servers easily. As a Visual Basic programmer, you can design custom functions for use in your reports by exposing them in ActiveX DLLs and registering the DLL on your system.

NOTE: Any development environment that allows the creation of COM-based Automation Servers can use the techniques similar to those described in this section. However, the code and examples shown here are based on Visual Basic.

The steps for creating automation servers in Visual Basic are slightly different, depending on the version of Visual Basic you are using.

Using Visual Basic 4.0

There are five primary steps to creating an Automation Server in Visual Basic 4.0 that exposes functions to the U2LCOM.DLL User Function Library:

1. *Set Up the Main Subroutine, Page 1089*
2. *Add a Class Module to the Project, Page 1089*
3. *Add User Functions to the Class Module, Page 1091*
4. *Name the Project, Page 1090*
5. *Compile the Project as an OLE DLL, Page 1090*

Set Up the Main Subroutine

When Visual Basic 4.0 first opens, it creates a default project and form for you. The entry point to your Automation Server DLL must be the Main subroutine. This subroutine needs to be defined in a Visual Basic Module, but can not be defined in a Class Module.

1. Remove the default form from your project by highlighting the form in the Project window and choosing the Remove File command from the File menu.
2. To add a new Module to the project, choose the Module command from the Insert menu.
3. In the Module window, add the following:

```
Sub Main()  
End Sub
```

You do not need to add any code to the Main subroutine, as long as it exists in your project.

Add a Class Module to the Project

Now you must add a Class Module to your project. The Class Module will contain any User Defined Functions that you wish to make available to the Seagate Crystal Reports Formula Editor.

4. To create a Class Module, choose the Class Module command from the Insert menu. A new Class Module window will appear, and the Class Module will be added to the Project window.
5. In the Properties window, set the following properties for the Class Module:

- **Instancing** = 2 - Creatable MultiUse
- **Name** = Any valid Class Module name
- **Public** = True

NOTE: For complete information on these properties, refer to Visual Basic Help.

If the Properties window is not visible, select the Class Module window, then choose the Properties command from the View menu.

Add User Functions to the Class Module

The next step is to add the actual User Defined Functions that you want to appear in the Seagate Crystal Reports Formula Editor. These functions are standard Visual Basic functions that you might define in any Visual Basic project. The only requirement is that the functions are declared Public.

6. Type in the following function:

```
Public Function DateToString(date1 As Date) As String
    DateToString = Format(date1, "Long Date")
End Function
```

NOTE: For complete information on how to define functions in Visual Basic, refer to your Visual Basic documentation.

The U2LCOM.DLL UFL supports most standard Visual Basic data types and arrays. For complete information on how the Seagate Crystal Reports Formula Editor interprets Visual Basic data types and arrays, see the sections *Visual Basic and Seagate Crystal Reports Variable Types, Page 1093*, and *Using Arrays, Page 1093*.

Name the Project

The U2LCOM.DLL UFL will only read Public functions exposed by Automation Servers named with a CRUFL prefix. For example, *CRUFLMyFunctions* is a valid project name for your Automation Server.

7. To change the name of your project in Visual Basic 4.0, choose the Options command from the Tools menu. The Options dialog box appears.
8. Click the Project Tab, and change the name of the project in the Project Name text box. The name should be CRUFLxxx, where xxx is a name of your choice.
9. While in the Options dialog box, make sure the Startup Form for the project is the Sub Main subroutine. Click OK when finished.

Compile the Project as an OLE DLL

10. Save the Module file, the Class Module file, and the Project file for your project.
11. Choose Make OLE DLL File from the File menu. Accept the default name for the DLL (your file name and location can be anything), and click OK. Visual Basic creates your Automation Server for you and registers it in your local system Registry.

Using Visual Basic 5.0

There are only four major steps to creating an Automation Server in Visual Basic 5.0. When you finish designing your Automation Server, the U2LCOM.DLL User Function Library will be able to access all of the functions that it exposes.

1. *Create a New ActiveX DLL Project, Page 1091*
2. *Add User Functions to the Class Module, Page 1091*
3. *Name the Project, Page 1091*
4. *Build the DLL, Page 1092*

Create a New ActiveX DLL Project

1. When you first run Visual Basic 5.0, the New Project dialog box appears. If you already have Visual Basic open, simply choose the New Project command from the File menu.
2. Double-click the ActiveX DLL icon in the New Project dialog box. Visual Basic creates a new project for you with a single Class Module.

NOTE: For complete information on creating an ActiveX DLL in Visual Basic 5.0, refer to your Visual Basic documentation.

Add User Functions to the Class Module

The next step is to add the actual User Defined Functions that you want to appear in the Seagate Crystal Reports Formula Editor. These functions are standard Visual Basic functions that you might define in any Visual Basic project. The only requirement is that the functions are declared Public.

3. Type in the following function:

```
Public Function DateToString(date1 As Date) As String  
    DateToString = Format(date1, "Long Date")  
End Function
```

The U2LCOM.DLL UFL supports most standard Visual Basic data types and arrays. For complete information on how the Seagate Crystal Reports Formula Editor interprets Visual Basic data types and arrays, see the sections *Visual Basic and Seagate Crystal Reports Variable Types, Page 1093*, and *Using Arrays, Page 1093*.

Name the Project

The U2LCOM.DLL UFL will only read Public functions exposed by Automation Servers named with a CRUFL prefix. For example, *CRUFLMyFunctions* is a valid project name for your Automation Server.

4. To change the name of your project in Visual Basic 5.0, highlight your project in the Project window, then change its *(Name)* property in the Properties window to CRUFLxxx, where xxx is a name of your choice.

Build the DLL

5. From the File menu, choose the Make CRUFLxxx.dll command. Once again, xxx is the name you chose. When this command executes, Visual Basic will compile your project into an ActiveX DLL (an Automation Server) and register it in your local system Registry.

Using the User Defined Functions

From Seagate Crystal Reports:

- 1 Create a new report and add tables to the report, or open an existing report.
- 2 From the Insert menu, choose Formula Field. The Insert Field dialog box appears.
- 3 Click New, and enter a name for the new formula in the Formula Name dialog box. Click OK and the Formula Editor appears.
- 4 Scroll down in the Functions list box to the Additional Functions section. Locate the User Defined Function you created.

User Defined Function names for functions created in Automation Servers are prefixed according to the project and class name used when you created the Automation Server. For instance, if you named your project *CRUFLProject*, named the class *Conversion*, and named the function *Square*, the User Defined Function would appear in the Formula Editor as *ProjectConversionSquare*.

- 5 Double-click the User Defined Function, and it appears in the Formula text box. Enter valid arguments for the function. For example:

```
ProjectConversionSquare(5)
```

- 6 Click Check. Make sure no errors appear in the function.
- 7 Click Accept, and place the formula in your report. Preview the report and verify that the function worked correctly.

Congratulations, you just created and used a User Defined Function.

Registering the Automation Server

An Automation Server must be registered on any system that it will be used on. Automation Servers are registered in the Windows 95 or Windows NT System Registry. The file can be physically stored anywhere on your hard drive because the Registry settings tell Windows where the file is located when it is needed.

When you make an ActiveX DLL project in Visual Basic 5.0 or an OLE DLL project in Visual Basic 4.0, using the Make command on the File menu, Visual Basic automatically registers the Automation Server on your system. Once registered, you can easily test and use the new User Defined Functions from the Seagate Crystal Reports Formula Editor.

Visual Basic 4.0 also includes an easy-to-use command-line application, REGSVR32.EXE, that will handle registering an Automation Server on your system. This application is located in the \CLISVR subdirectory of the directory in which you installed Visual Basic. This application can be used from an MS-DOS prompt by simply specifying the name of the Automation Server DLL. For example:

```
RegSvr32.exe C:\Projects\CRUFLMyFunctions.dll
```

Visual Basic and Seagate Crystal Reports Variable Types

Seagate Crystal Reports will support most common Visual Basic data types provided through a User Defined Function developed in Visual Basic. The following table shows how Seagate Crystal Reports converts the most common Visual Basic data types to data types supported by the Formula Editor:

<i>Visual Basic Data Types</i>	<i>Formula Editor Data Types</i>
Integer	NumberVar
Long	
Single	
Double	
Currency	CurrencyVar
Date	DateVar
Boolean	BooleanVar
String	StringVar

Using Arrays

Arrays can be passed to any User Defined Function as a parameter of the function. This means that when you design your function in Visual Basic, the function can accept an array of values of any of the supported data types. However, the function can not return an array to the Seagate Crystal Reports Formula Editor. The following Visual Basic function is an acceptable User Defined Function for Seagate Crystal Reports:

```
Public Function GetNthItem (sArray() As String, n As Integer) As String
    GetNthItem = sArray(n)
End Function
```

Error Handling

If it is possible for your User Defined Function to produce an error, the Seagate Crystal Reports Formula Editor should be notified of the error. Many types of errors are automatically handled by the Formula Editor. Passing the wrong type of data to a function, for instance, will be recognized and trapped by Seagate Crystal Reports. However, if you design a function that can produce an error unique to that function, you should provide a means for reporting that error to the Formula Editor.

To send error messages to the Formula Editor, define the UFErrorText string property in your Class Module. This property should be defined Public, using code similar to this:

```
Public UFErrorText As String
```

Any time you trap for an error condition, simply set the UFErrorText property to the error text you want reported in Seagate Crystal Reports. Setting the value of this property triggers the error in Seagate Crystal Reports, and Seagate Crystal Reports displays a dialog box containing the error message that you assigned to UFErrorText.

NOTE: You should not use the UFErrorText property for anything other than returning errors from User Defined Functions. The U2LCOM.DLL UFL regularly resets the value of this property, so data can be lost if stored in UFErrorText for any reason other than reporting an error.

Special Purpose Functions

You can define several special purpose functions in your Class Module in addition to the User Defined Functions that you are creating. The Seagate Crystal Reports Formula Editor can look for and process code defined in any of the following functions:

```
Public Function UFInitialize () As Long  
Public Function UFTerminate () As Long  
Public Sub UFStartJob (job As Long)  
Public Sub UFEndJob (job As Long)
```

Be sure to define the functions in your code exactly as they appear above. If not defined correctly, they will be ignored by Seagate Crystal Reports. These functions are completely optional when creating your Visual Basic Automation Server. They are provided to assist you with the design of your User Defined Functions.

UFInitialize

This function is called just after the DLL is loaded into memory. Use this function to handle one-time initialization of variables.

Return a value of 0 (zero) to indicate successful initialization. Any non-zero value indicates initialization failed.

UFTerminate

This function is called just before the DLL is unloaded from memory. Use this function to clean up any allocated memory or other data before unloading the DLL.

Return a value of 0 (zero) when finished cleaning up memory.

UFStartJob

This procedure is called by Seagate Crystal Reports just before User Defined Functions are evaluated (or reevaluated). The Seagate Crystal Reports job number is passed to the function as the only parameter. Use this function to handle any initialization on a per-job basis.

UPEndJob

This procedure is called by Seagate Crystal Reports when the current job finishes, which happens when all pages of a report have been generated, before UFStartJob is called again, or before UFTerminate is called. This function also accepts the Seagate Crystal Reports job number as its only parameter. Handle any clean-up necessary on a per-job basis using this function.

Distribution

The easiest way to distribute any Visual Basic project is to use the Setup Wizard to create an installation program. In addition to making sure all necessary files are installed where appropriate, the Setup Wizard can add code to your installation to register the Automation Server in the user's system Registry.

In addition to installing and registering your Automation Server, you must also provide the U2LCOM.DLL UFL file on any system that will use the functions exposed by your Automation Server. If the system has Seagate Crystal Reports installed, then this file will already be located in the system's \WINDOWS\CRYSTAL directory.

Sample UFL Automation Server

Seagate Crystal Reports includes the source code for a sample automation server that exposes a User Defined Function. The code includes a Visual Basic 4.0 project file that can be compiled in either Visual Basic 4.0 or 5.0. Use this sample as a reference for building your own Automation Server based User Defined Functions. You can even copy the Class Module provided into your own projects as a head-start to building Automation Servers for U2LCOM.DLL.

Additional Information

- **32-bit Support Only:** U2LCOM.DLL is a 32-bit UFL only and, therefore, supports only 32-bit Automation Server DLLs. To create User Defined Functions for Seagate Crystal Reports, you must have the 32-bit edition of Visual Basic 4.0 or Visual Basic 5.0 (or another 32-bit development environment that supports the creation of COM-based Automation Servers).
- **Ranges:** The range data type available in the Seagate Crystal Reports Formula Editor is not supported in COM-based User Defined Functions.
- **Reserved Names:** Certain names are reserved and can not be used as User Defined Function names. The following names are reserved by the Seagate Crystal Reports Formula Editor for special purposes:
 - UFInitialize
 - UFTerminate
 - UFStartJob
 - UPEndJob

For more explanation of these function names, see *Special Purpose Functions, Page 1094*. In addition, User Defined Functions can not use the same name as any of the functions exposed by the IDispatch interface used by COM:

- QueryInterface
- AddRef
- Release
- GetTypeInfoCount
- GetTypeInfo
- GetIDsOfNames
- Invoke

- **Function Name Prefixing:** To ensure a unique name when User Defined Functions appear in the Formula Editor, Seagate Crystal Reports appends a prefix to each function name that is generated from the project and Class Module names in the original source code. The first part of the prefix is the project name without the CRUFL prefix. The rest of the function name prefix is the Class Module name.

Once the prefix for the function name is generated, all non-alphanumeric characters are removed, and the prefix is applied to the original function name. The following table illustrates this process:

Project Name:	CRUFLTestFunctions
Class Module Name:	Conversion
User Defined Function Name:	Date_To_String()
Formula Editor Function:	TestFunctionsConversionDateToString()

This function name prefixing can be turned off if you are sure that your function names will not conflict with any other function names recognized by the Formula Editor. To turn off function name prefixing:

1. Define a Boolean property for the class called UFPrefixFunctions.
2. Set the value of the property to False in the Initialize subroutine for the class.

For example:

```
Public UFPrefixFunctions As Boolean

Private Sub Class_Initialize()
    UFPrefixFunctions = False
End Sub
```

NOTE: *Function name prefixing is designed to eliminate function name conflicts. If you turn off function name prefixing and your function name conflicts with another function, you may get unpredictable results.*

- **ByRef vs. ByVal:** Arguments can be passed to User Defined Functions written in Visual Basic either ByRef or ByVal. ByRef is the default method for Visual Basic, but both methods will work. For instance, all of the following functions are valid:

```
Public Function Test1 (num1 As Integer) As Integer  
Public Function Test1 (ByRef num1 As Integer) As Integer  
Public Function Test1 (ByVal num1 As Integer) As Integer
```

15

Creating User Defined Functions in Delphi 3.0

What you will find in this chapter...

Using Delphi 3.0, Page 1101

Using the User Defined Functions, Page 1102

Delphi and Seagate Crystal Reports Data Types, Page 1103

Using Arrays, Page 1103

Error Handling, Page 1104

Special Purpose Functions, Page 1104

Sample UFL, Page 1105

This topic is only supported by the Professional Edition of Seagate Crystal Reports.

The Seagate Crystal Reports Formula Editor and formula language are powerful tools, enabling you to do a wide variety of report-related tasks easily and efficiently. The formula language is expandable as well. That is, while it already includes a large selection of useful functions, it also comes with the ability to accept new functions that you define to meet your needs.

User Defined Functions that are recognized by the Seagate Crystal Reports Formula Editor can be created in a Dynamic Link Library or, for 32-bit environments, in an Automation Server. This section demonstrates how to create User Defined functions in an Automation Server using Borland Delphi 3.0. For information on how to create User Defined Functions in a Dynamic Link Library using the C or C++ programming language, see *Creating User Defined Functions, Page 1065*. For information on how to create User Defined Functions in an Automation Server using Visual Basic, see *Creating User Defined Functions in Visual Basic, Page 1088*.

Creating User Defined Functions in Delphi

The Seagate Crystal Reports Formula Editor can access User Defined Functions through a User Function Library (UFL). A User Function Library is a specially designed Dynamic Link Library that exposes one or more functions that you create. UFLs can be designed and programmed in any language that supports the development of Windows DLLs.

The 32-bit version of Seagate Crystal Reports Professional Edition includes the User Function Library U2LCOM.DLL. This UFL is installed in your \WINDOWS\CRYSTAL directory when you install Seagate Crystal Reports and provides an interface through which you can expose User Defined Functions in Automation Servers.

An Automation Server is a Dynamic Link Library or executable file that exposes its functionality to other modules and processes through the Component Object Model. For complete information on the Component Object Model (COM), refer to Microsoft documentation or the Microsoft World Wide Web site. U2LCOM.DLL is a UFL that can access any functions exposed by any Automation Servers named with a CRUFL prefix. This means that you can create an Automation Server in any language environment that supports COM, name the server with a CRUFL prefix, register the server on a system, and the Seagate Crystal Reports Formula Editor will access and make available any functions exposed by that Automation Server.

NOTE: You may be more familiar with Automation Servers as OLE Automation. Though the technology behind Automation Servers is the same as OLE Automation, the correct name for these servers is now simply Automation Servers.

Delphi 3.0 allows you to design 32-bit Automation Servers easily. As a Delphi programmer, you can design custom functions for use in your reports by exposing them in an ActiveX Library and registering the library (Automation Server) on your system.

Using Delphi 3.0

NOTE: Version 3.0 of Delphi is required to create Automation Servers containing User Defined Functions.

There are six primary steps to creating User Defined Functions in an automation server in Delphi 3.0:

3. *Create the Project, Page 1101*
4. *Create the Automation Object, Page 1101*
5. *Add Methods to the Type Library, Page 1101*
6. *Register the Type Library, Page 1102*
7. *Create the User Defined Functions, Page 1102*
8. *Build the Project, Page 1102*

Create the Project

When Delphi first opens, it creates a default project and form for you.

1. Choose the New command from the File menu.
2. Click the ActiveX Tab in the New Items dialog box, and double-click the ActiveX Library icon. Delphi creates a default ActiveX Library for you.

The U2LCOM.DLL UFL will only read functions exposed by Automation Servers named with a CRUFL prefix. For example, CRUFLMyFunctions is a valid project name for your Automation Server.

3. Choose the Save Project As command from the File menu, and save your Delphi project. The project should be named CRUFLxxx.DPR, where xxx is a name of your choice.

Create the Automation Object

4. Choose the New command from the File menu again.
5. Click the ActiveX Tab in the New Items dialog box, and double-click the Automation Object icon. The Automation Object Wizard appears.
6. Enter a class name appropriate to the functions you will create. Make sure Instancing is set to Multiple Instance, and click OK. The Type Library Editor appears.
7. Make sure the name of the type library for your project matches the project name you created earlier. If not, change the name of the type library to CRUFLxxx, where xxx is the name you chose.

Add Methods to the Type Library

The methods in the class you specified when you created the type library will become User Defined Functions that appear in the Seagate Crystal Reports Formula Editor.

8. Right-click the interface for your type library. The interface name is identical to the class name you specified preceded by an I.

9. Choose New from the menu that appears, and choose Method. A new method appears below the interface in the Object list pane.
10. Name the method according to the function you want to create.
11. On the Attributes Tab for the method, declare your function. For example:

```
function Square (Number: double): double;
```

NOTE: For complete information on how to declare functions in Delphi, refer to your Delphi documentation.

The U2LCOM.DLL UFL supports most standard Delphi data types and arrays. For complete information on how the Seagate Crystal Reports Formula Editor interprets Delphi data types and arrays, see *Delphi and Seagate Crystal Reports Data Types, Page 1103*.

Continue creating methods for all User Defined Functions you want to appear in the Seagate Crystal Reports Formula Editor.

Register the Type Library

12. Click Register in the Type Library Editor. Delphi creates your type library and registers it on your system. A message should appear indicating that the ActiveX automation server was successfully registered. Click OK in the message box.

NOTE: If the type library is not registered successfully on your system, create the type library and object methods over again. If you continue to have problems, refer to your Delphi documentation on creating type libraries.

Create the User Defined Functions

13. Minimize the Type Library Editor, and locate the declaration of the method you created in the your type library in the Unit1.pas unit.
14. Enter code for the function as desired. For example:

```
function TConversion.Square(Number: Double): Double;
begin
  result := Number * Number;
end;
```

Continue coding all methods that you declared for the interface.

Build the Project

15. Save all files in your project, and choose Build All from the Project menu. Delphi builds your automation server, and the methods you declared are now available from the Seagate Crystal Reports Formula Editor.

Using the User Defined Functions

From Seagate Crystal Reports:

- 1 Create a new report and add tables to the report, or open an existing report.

- 2 From the Insert menu, choose Formula Field. The Insert Field dialog box appears.
- 3 Click New, and enter a name for the new formula in the Formula Name dialog box. Click OK and the Formula Editor appears.
- 4 Scroll down in the Functions list box to the Additional Functions section. Locate the User Defined Function you created.

User Defined Function names for functions created in Automation Servers are prefixed according to the project and class name used when you created the automation server. For instance, if you named your project *CRUFLProject*, named the class *Conversion*, and named the function *Square*, the User Defined Function would appear in the Formula Editor as *ProjectConversionSquare*.

- 5 Double-click the User Defined Function, and it appears in the Formula text box. Enter valid arguments for the function. For example:

```
ProjectConversionSquare(5)
```

- 6 Click Check. Make sure no errors appear in the function.
- 7 Click Accept, and place the formula in your report. Preview the report and verify that the function worked correctly.

Congratulations, you just created and used a User Defined Function.

Delphi and Seagate Crystal Reports Data Types

Seagate Crystal Reports will support most common Delphi data types provided through a User Defined Function developed in Delphi 3.0. The following table shows how Seagate Crystal Reports converts the most common Delphi data types to data types supported by the Formula Editor:

<i>Delphi Data Types</i>	<i>Formula Editor Data Types</i>
ShortInt	NumberVar
Integer	
LongInt	
Real	
Single	
Double	
Currency	CurrencyVar
Date	DateVar
Boolean	BooleanVar
String	StringVar

Using Arrays

Arrays can be passed to any User Defined Function as a parameter of the function. This means that when you design your function in Delphi, the function can accept an array of values of any of the supported data types. However, the function can not return an array to the Seagate Crystal Reports

Formula Editor. The following Delphi function is an acceptable User Defined Function for Seagate Crystal Reports:

```
Function GetNthItem (A: MyArray, n: Integer): Integer;
begin
  GetNthItem := A[n];
End;
```

Error Handling

If it is possible for your User Defined Function to produce an error, the Seagate Crystal Reports Formula Editor should be notified of the error. Many types of errors are automatically handled by the Formula Editor. Passing the wrong type of data to a function, for instance, will be recognized and trapped by Seagate Crystal Reports. However, if you design a function that can produce an error unique to that function, you should provide a means for reporting that error to the Formula Editor.

To send error messages to the Formula Editor, define the UFErrorText string property in your Class Module. This property should be defined Public, using code similar to this:

```
Property UFErrorText: String;
```

Any time you trap for an error condition, simply set the UFErrorText property to the error text you want reported in Seagate Crystal Reports. Setting the value of this property triggers the error in Seagate Crystal Reports, and Seagate Crystal Reports displays a dialog box containing the error message that you assigned to UFErrorText.

NOTE: You should not use the UFErrorText property for anything other than returning errors from User Defined Functions. The U2LCOM.DLL UFL regularly resets the value of this property, so data can be lost if stored in UFErrorText for any reason other than reporting an error.

Special Purpose Functions

You can define several special purpose functions in your Class Module in addition to the User Defined Functions that you are creating. The Seagate Crystal Reports Formula Editor can look for and process any of the following class methods:

```
Function UFInitialize: Integer;
Function UFTerminate: Integer;
Procedure UFStartJob (job: Integer)
Procedure UFEndJob (job: Integer)
```

Be sure to declare the methods in your code exactly as they appear above. If not declared correctly, they will be ignored by Seagate Crystal Reports. These methods are completely optional when creating your Delphi Automation Server. They are provided to assist you with the design of your User Defined Functions.

UInitialize

This function is called just after the DLL is loaded into memory. Use this function to handle one-time initialization of variables.

Return a value of 0 (zero) to indicate successful initialization. Any non-zero value indicates initialization failed.

UFTerminate

This function is called just before the DLL is unloaded from memory. Use this function to clean up any allocated memory or other data before unloading the DLL.

Return a value of 0 (zero) when finished cleaning up memory.

UFStartJob

This procedure is called by Seagate Crystal Reports just before User Defined Functions are evaluated (or reevaluated). The Seagate Crystal Reports job number is passed to the function as the only parameter. Use this function to handle any initialization on a per-job basis.

UFEndJob

This procedure is called by Seagate Crystal Reports when the current job finishes, which happens when all pages of a report have been generated, before UFStartJob is called again, or before UFTerminate is called. This function also accepts the Seagate Crystal Reports job number as its only parameter. Handle any clean-up necessary on a per-job basis using this function.

Sample UFL

A sample UFL project is located on the Seagate Crystal Reports installation CD. The self-extracting executable file UFLDELPH.EXE is located in the \SAMPAPPS\COMUFLS\DELPHI directory on the CD. Simply run the executable file to install the sample project and all related files on your hard drive. This Delphi project creates three User Defined Functions that can be used in the Seagate Crystal Reports Formula Editor.

Additional Information

- **32-bit Support Only:** U2LCOM.DLL is a 32-bit UFL only and, therefore, supports only 32-bit Automation Server DLLs. To create User Defined Functions for Seagate Crystal Reports, you must have Delphi 3.0 (or another 32-bit development environment that supports the creation of COM-based Automation Servers).
- **Ranges:** The range data type available in the Seagate Crystal Reports Formula Editor is not supported in COM-based User Defined Functions.

- **Reserved Names:** Certain names are reserved and can not be used as User Defined Function names. The following names are reserved by the Seagate Crystal Reports Formula Editor for special purposes:

 - UFInitialize
 - UFTerminate
 - UFStartJob
 - UFEnderJob

For more explanation of these function names, see *Special Purpose Functions, Page 1104*. In addition, User Defined Functions can not use the same name as any of the functions exposed by the IDispatch interface used by COM:

 - QueryInterface
 - AddRef
 - Release
 - GetTypeInfoCount
 - GetTypeInfo
 - GetIDsOfNames
 - Invoke

- **Function Name Prefixing:** To ensure a unique name when User Defined Functions appear in the Formula Editor, Seagate Crystal Reports appends a prefix to each function name that is generated from the project and class names. The first part of the prefix is the project name without the CRUFL prefix. The rest of the function name prefix is the class name.

Once the prefix for the function name is generated, all non-alphanumeric characters are removed, and the prefix is applied to the original function name. The following table illustrates this process:

Project Name:	CRUFLTestFunctions
Class Name:	Conversion
User Defined Function Name:	Date_To_String()
Formula Editor Function:	TestFunctionsConversionDateToString()

- **By reference vs. by value:** Arguments can be passed to User Defined Functions written in Delphi 3.0, either by value or by reference. For instance, both of the following functions are valid:

```
Function Test1 (num1: Integer): Integer;
Function Test1 (var num1: Integer): Integer;
```

I N D E X

A	D	E	X
Access Notes Sparse Array Properties tutorial, VCL 837	EMailIMBCCList 509 ExchangeFolder 510 ExchangePassword 511 ExchangeProfile 511 Formulas 512 GraphData 513 GraphOptions 515 GraphText 517 GraphType 518 GroupCondition 520 GroupSelectionFormula 522 GroupSortFields 523 LastErrorNumber 524 LastErrorString 525 LogOnInfo 526 MarginBottom 528 MarginLeft 529 MarginRight 529 MarginTop 530 ParameterFields 531 Password 532 PrintDay 533 PrinterCollation 534 PrinterCopies 535 PrinterDriver 536 PrinterName 537 PrinterPort 538 PrinterStartPage 539 PrinterStopPage 540 PrintFileCharSepQuote 541 PrintFileCharSepSeparator 542 PrintFileLinesPerPage 542 PrintFileName 543 PrintFileODBCPassword 544 PrintFileODBCSource 545 PrintFileODBCTable 545 PrintFileODBCUser 546 PrintFileType 547 PrintFileUseRptDateFmt 549 PrintFileUseRptNumberFmt 550 PrintMonth 551 PrintYear 552 ProgressDialog 553 RecordsPrinted 554 RecordsRead 555 RecordsSelected 555 Report Title 560 ReportDisplayPage 556 ReportFileName 557 ReportLatestPage 558 ReportSource 559 ReportStartPage 559 SectionFont 561	SectionFormat 562 SectionLineHeight 564 SectionMinHeight 565 SelectionFormula 566 SessionHandle 567 SortFields 568 SQLQuery 569 Status 570 StoredPracParam 571 SubreportToChange 572 UserName 574 WindowAllowDrillDown 574 WindowBorderStyle 575 WindowControlBox 576 WindowControls 577 WindowHeight 578 WindowLeft 578 WindowMaxButton 579 WindowMinButton 580 WindowParentHandle 581 WindowShowCancelBtn 582 WindowShowCloseBtn 583 WindowShowExportBtn 583 WindowShowGroupTree 584 WindowShowNavigationCtrls 585 WindowShowPrintBtn 586 WindowShowPrintSetupBtn 586 WindowShowProgressCtrls 587 WindowShowRefreshBtn 588 WindowShowSearchBtn 589 WindowShowZoomCtl 590 WindowState 590 WindowTitle 591 WindowTop 592 WindowWidth 593	ActiveX, Crystal Smart Viewer 34 ActiveX, Design-time Control 26 activity reports, web designing 20 requesting from browser 21 Add method, REOL 707 AddGroup method, REOL 689 adding the ActiveX Control to project 83 adding VCL to project 108 administration Crystal Web Report Server... 10 API Report Engine 48 API, Report Engine using in Visual Basic 78

A
 Application Object
 CanClose method, REOL ... 612
 ClearError method, REOL...613
 LogOffServer method,
 REOL613
 LogOnServer method,
 REOL614
 OpenReport method,
 REOL614
 Application Object, REOL.....611
 applications
 Report Engine76
 Area Object, REOL.....615
 AreaOptions Object, REOL617
 Areas Collection, REOL619
 AuthentiCode certification.....35
 Automation Server.....26

B

before using the Report Engine...45
 BlobField Object, REOL620
 BoundReportFooter property497
 BoundReportHeading
 property497
 Box Object, REOL.....621

C

C syntax
 DEVMODE.....353
 PECancelPrintJob121
 PECanCloseEngine.....122
 PCheckFormula123
 PCheckGroupSelection-
 Formula124
 PCheckSelection-
 Formula126
 PECloseButtonClickedEvent-
 Info structure.....391
 PECloseEngine127
 PEClosePrintJob129
 PECloseSubreport130
 PECloseWindow131
 PEConvertPFIInfoVInfo....133
 PEConvertVInfo to PFIInfo....135
 PEDeleteNthGroupSort-
 Field.....136
 PEDeleteNthSortField138
 PEDiscardSavedData140
 PEDrillOnDetailEvent-
 Info392
 PEDrillOnGroupEvent-
 Info394
 PEEnableEvent141
 PEEnableEventInfo397
 PEEnableProgressDialog ...142

PEExportOptions399
 PEExportPrintWindow144
 PEExportTo.....146
 PEFieldValueInfo407
 PEGeneralPrintWindowEvent-
 Info.....409
 PEGetAreaFormat.....147
 PEGetAreaFormat-
 Formula.....148
 PEGetEnableEventInfo151
 PEGetErrorCode152
 PEGetErrorText153
 PEGetExportOptions.....155
 PEGetFormula156
 PEGetGraphData158
 PEGetGraphOptions.....159
 PEGetGraphText161
 PEGetGraphType162
 PEGetGroupCondition164
 PEGetGroupOptions167
 PEGetGroupSelection-
 Formula168
 PEGetHandleString.....170
 PEGetJobStatus171
 PEGetMargins173
 PEGetMinimumSection-
 Height175
 PEGetNDetailCopies176
 PEGetNFormulas177
 PEGetNGroups179
 PEGetNGroupSortFields180
 PEGetNPages181
 PEGetNParameterFields....183
 PEGetNParams184
 PEGetNSections185
 PEGetNSortFields.....187
 PEGetNSubreportsIn-
 Section188
 PEGetNTables189
 PEGetNthFormula191
 PEGetNthGroupSort-
 Field193
 PEGetNthParam195
 PEGetNthParameterField ...197
 PEGetNthParamInfo199
 PEGetNthSortField200
 PEGetNthSubreportIn-
 Section202
 PEGetNthTableLocation203
 PEGetNthTableLogOn-
 Info205
 PEGetNthTableSession-
 Info.....206
 PEGetNthTableType207
 PEGetPrintDate209
 PEGetPrintOptions211
 PEGetReportTitle212, 214
 PEGetSectionCode.....215
 PEGetSectionFormat217
 PEGetSectionFormat-
 Formula219
 PEGetSelectedPrinter222
 PEGetSelectionFormula224
 PEGetSQLQuery226
 PEGetSubreportInfo228
 PEGetTrackCursorInfo.....229
 PEGetVersion.....231
 PEGetWindowHandle.....232
 PEGetWindowOptions.....234
 PEGraphDataInfo410
 PEGraphOptions.....413
 PEGraphTextInfo.....415
 PEGroupOptions.....418
 PEGroupTreeButtonClicked-
 EventInfo421
 PEHasSavedData237
 PEIsPrintJobFinished238
 PEJobInfo423
 PELogOffServer.....240
 PELogOnInfo425
 PELogOnServer.....241
 PELogOnSQLServerWith-
 PrivateInfo.....243
 PENextPrintWindow-
 Magnification.....245
 PEOpen Engine.....247
 PEOpenPrintJob248
 PEOpenSubreport250
 PEOOutputToFile251
 PEOOutputToPrinter.....254
 PEOOutputToWindow.....257
 PEParameterFieldInfo427
 PEParameterInfo432
 PEPrintControlsShowing ...261
 PEPrintOptions435
 PEPrintReport.....263
 PEPrintWindow265
 PEReadingRecordsEvent-
 Info438
 PEReportSummaryInfo440
 PESearchButtonClicked-
 EventInfo.....442
 PESectionOptions444
 PESelectPrinter.....267
 PESessionInfo.....448
 PESetAreaFormat269
 PESetAreaFormatFormula...271
 PESetDialogParent-
 Window274
 PESetEventCallback275
 PESetFont.....282
 PESetFormula.....287

PESetGraphData	289	UXDPostFolderOptions.....	479	CRPE Class Library constructor
PESetGraphOptions	291	UXDSMIOptions	478	CRPEEngine::CRPEEngine ..
PESetGraphText	292	UXDVIMOptions	481	840
PESetGraphType	294	UXFCharSeparated-		CRPEExportOptions::
PESetGroup	296	Options	482	CRPEExportOptions
PESetGroupOptions	302	UXFCommaTabSeparated-		932
PESetGroupSelection-		Options	484	CRPEGraphDataInfo::
Formula	303	UXFDIFOptions	485	CRPEGraphDataInfo
PESetMargins	305	UXFHTML3Options	487	936
PESetMinimumSection-		UXFODBCOptions.....	488	CRPEGraphOptions::
Height.....	307	UXFPaginatedText-		CRPEGraphOptions
PESetNDetailCopies	309	Options	489	938
PESetNthGroupSortField	311	UXFRecordStyleOptions....	490	CRPEGraphTextInfo::
PESetNthParam	313	CancelButtonClicked event,		CRPEGraphTextInfo
PESetNthParameterField	315	REOL.....	725	940
PESetNthSortField	316	CancelPrinting method,		CRPEJob::CRPEJob
PESetNthTableLocation	319	REOL	690	852
PESetNthTableLogOn-		CanClose method, REOL	612	CRPEJobInfo::
Info	320	ChangeTableLocation method,		CRPEJobInfo
PESetNthTableSession-		VCL	824	930
Info	322	changing properties for ActiveX		CRPELogOnInfo::
PESetPrintDate	324	Control	84	CRPELogOnInfo
PESetPrintOptions	326	changing properties in the		943
PESetReportSummary-		VCL	110	CRPEParameterFieldInfo::
Info	327	changing selection formulas in web		CRPEParameterField-
PESetReportTitle	329	reports	17	Info
PESetSectionFormat	331	Check method, REOL	649	945
PESetSectionFormat-		classes		CRPEParameterInfo::
Formula	333	REC Library	111	CRPEParameterInfo
PESetSelectionFormula	336	Clear method, VCL	825	947
PESetSQLQuery	337	ClearError method,		CRPEPrintOptions::
PESetTrackCursorInfo	339	REOL	613, 690	CRPEPrintOptions
PESetWindowOptions	341	Close method,		948
PEShowFirstPage	342	REOL.....	720, 722, 724	CRPESectionOptions::
PEShowGroupEventInfo	450	Close Window method, VCL	825	CRPESectionOptions
PEShowLastPage	342	CloseButtonClicked event,		951
PEShowNextPage	342	REOL.....	726	CRPESessionInfo::
PEShowNthPage	345	ClosePrintWindow event,		CRPESessionInfo
PEShowPreviousPage	342	REOL.....	726	953
PEShowPrintControls	346	COLORREF structure	353	CRPESubreportInfo::
PESTartEventInfo	451	Connect property.....	498	CRPESubreportInfo
PESTartPrintJob	348	Connect property, VCL.....	741	954
PEStopEventInfo	453	controls		CRPETableLocation::
PESubreportInfo	455	ActiveX	82	CRPETableLocation
PETableLocation	457	CopiesToPrinter property	500	955
PETableType	459	CopiesToPrinter property,		CRPETableType::
PETestNthTable-		VCL	742	CRPETableType
Connectivity	349	CreatePageGenerator method,		956
PETrackCursorInfo	461	REOL.....	665	CRPEEngine::
PEValueInfo	466	CrossTabObject Object,		GetClose
PEWindowOptions	470	REOL.....	622	840
PEZoomLevelChanging-		CRPE Class Library class		GetClose
EventInfo	473	CRPEJob::public CObject....	849	841
PEZoomPreviewWindow	351	CRPEngine:		GetClose
UXDDiskOptions	475	public CObject.....	956	843
UXDMAPIOptions	476			GetClose

CRPEJob::CheckFormula	850
CRPEJob::CheckGroup- SelectionFormula	851
CRPEJob::: CheckSelectionFormula ...	851
CRPEJob::Close	852
CRPEJob::CloseWindow	852
CRPEJob::DeleteNthGroup- SortField.....	853
CRPEJob::: DeleteNthSortField.....	854
CRPEJob::: DiscardSavedData.....	854
CRPEJob::: EnableProgressDialog.....	855
CRPEJob::: ExportPrintWindow.....	855
CRPEJob::ExportTo.....	856
CRPEJob::GetErrorCode	857
CRPEJob::GetErrorText.....	857
CRPEJob::GetFormula	858
CRPEJob::GetGraphData....	858
CRPEJob::: GetGraphOptions	860
CRPEJob::GetGraphText	861
CRPEJob::GetGraphType ...	862
CRPEJob::: GetGroupCondition	864
CRPEJob::GetGroupSelection- Formula	868
CRPEJob::GetJobHandle ...	868
CRPEJob::GetJobStatus.....	869
CRPEJob::GetMargins	870
CRPEJob::GetMinimum- SectionHeight	870
CRPEJob::: GetNDetailCopies.....	872
CRPEJob::GetNFormulas... <td>872</td>	872
CRPEJob::GetNGroups.....	873
CRPEJob::: GetNGroupSortFields....	873
CRPEJob::GetNPages	874
CRPEJob::: GetNParameterFields	874
CRPEJob::GetNParams.....	875
CRPEJob::GetNSections....	875
CRPEJob::GetNSortFields ...	876
CRPEJob::: GetNSubreportsIn- Section	876
CRPEJob::GetNTables	877
CRPEJob::GetNth- Formula	877
CRPEJob::: GetNthGroupSortField	878
CRPEJob::GetNthParam	879
CRPEJob::: GetNthParameterField	880
CRPEJob::: GetNthParamInfo	880
CRPEJob::: GetNthSortField.....	881
CRPEJob::: GetNthSubreportInSection	882
CRPEJob::: GetNthTableLocation	883
CRPEJob::: GetNthTableLogOnInfo ...	883
CRPEJob::GetNthTableSession- Info.....	884
CRPEJob::: GetNthTableType	885
CRPEJob::GetPrintDate.....	885
CRPEJob::: GetPrintOptions	886
CRPEJob::GetReportTitle ...	887
CRPEJob::: GetSectionCode	887
CRPEJob::: GetSectionFormat.....	888
CRPEJob::: GetSelectedPrinter.....	889
CRPEJob::: GetSelectionFormula	890
CRPEJob::GetSQLQuery	891
CRPEJob::: GetSubreportInfo	891
CRPEJob::: GetWindowHandle	892
CRPEJob::HasSavedData ...	892
CRPEJob::IsJobFinished	893
CRPEJob::NextWindow- Magnification	894
CRPEJob::: OpenSubreportJob	894
CRPEJob::: OutputToPrinter	895
CRPEJob::: OutputToWindow	895
CRPEJob::: PrintControlsShowing	897
CRPEJob::PrintWindow	897
CRPEJob::SelectPrinter	898
CRPEJob::SetDialogParent- Window	899
CRPEJob::SetFont	899
CRPEJob::SetFormula	903
CRPEJob::SetGraphData ...	903
CRPEJob::SetGraphText....	906
CRPEJob::: SetGroupCondition	909
CRPEJob::SetGroupSelection- Formula	912
CRPEJob::SetMargins	912
CRPEJob::SetMinimum- SectionHeight	913
CRPEJob::: SetNDetailCopies.....	914
CRPEJob::: SetNthGroupSortField	915
CRPEJob::SetNthParam	916
CRPEJob::: SetNthParameterField.....	917
CRPEJob::: SetNthSortField	918
CRPEJob::: SetNthTableLocation.....	919
CRPEJob::: SetNthTableLogOnInfo....	919
CRPEJob::: SetNthTableSessionInfo...	920
CRPEJob::SetPrintDate	921
CRPEJob::: SetPrintOptions	922
CRPEJob::SetReportTitle	922
CRPEJob::: SetSectionFormat	923
CRPEJob::: SetSelectionFormula.....	924
CRPEJob::SetSQLQuery	925
CRPEJob::ShowFirstPage ...	925
CRPEJob::ShowLastPage ...	926
CRPEJob::ShowNextPage	926
CRPEJob::: ShowPreviousPage	927
CRPEJob::: ShowPrintControls	928
CRPEJob::Start.....	928
CRPEJob::TestNthTable- Connectivity.....	929
CRPEJob::ZoomPreview- Window	929
CRPE Class Library structure	
CRPEExportOptions	931
CRPEGraphDataInfo	935
CRPEGraphOptionsInfo	937
CRPEGraphTextInfo	939
CRPEJobInfo	941
CRPELogOnInfo	942
CRPEParameterFieldInfo ...	943
CRPEParameterInfo	947
CRPEPrintOptions	948
CRPESectionOptions	949
CRPESessionInfo	952
CRPESubreportInfo	954
CRPETableLocation	954
CRPETableType	955

CRPEEngine::CanClose method, CRPE Class Library	840	CRPEJob::CRPEJob constructor, NewEra Class Library	972
CRPEEngine::Close method, CRPE Class Library	840	CRPEJob::DeleteNthGroupSortField method, CRPE Class Library	853
CRPEEngine::CRPEEngine constructor, CRPE Class Library	840	CRPEJob::DeleteNthGroupSortField method, NewEra Class Library	972
CRPEEngine::GetEngine method, CRPE Class Library	841	CRPEJob::DeleteNthSortField method, CRPE Class Library	854
CRPEEngine::GetEngineStatus method, CRPE Class Library	841	CRPEJob::DeleteNthSortField method, NewEra Class Library	973
CRPEEngine::GetErrorCode method, CRPE Class Library	841	CRPEJob::DiscardSavedData method, CRPE Class Library	854
CRPEEngine::GetErrorText method, CRPE Class Library	842	CRPEJob::EnableProgressDialog method, CRPE Class Library	855
CRPEEngine::GetNPrintJobs method, CRPE Class Library	843	CRPEJob::ExportPrintWindow method, CRPE Class Library	855
CRPEEngine::GetVersion method, CRPE Class Library	843	CRPEJob::ExportPrintWindow method, NewEra Class Library	974
CRPEEngine::LogOffServer method, CRPE Class Library	844	CRPEJob::ExportTo method, CRPE Class Library	856
CRPEEngine::LogOnServer method, CRPE Class Library	845	CRPEJob::ExportTo method, NewEra Class Library	974
CRPEEngine::LogOnSQLServer-WithPrivateInfo method, CRPE Class Library	846	CRPEJob::ExportTo method, NewEra Class Library	975
CRPEEngine::Open method, CRPE Class Library	847	CRPEJob::GetErrorCode method, CRPE Class Library	857
CRPEEngine::OpenJob method, CRPE Class Library	847	CRPEJob::GetErrorCode method, NewEra Class Library	975
CRPEEngine::PrintReport method, CRPE Class Library	848	CRPEJob::GetErrorText method, CRPE Class Library	857
CRPEExportOptions class, NewEra Class Library	1043	CRPEJob::GetErrorText method, NewEra Class Library	975
CRPEExportOptions, CRPE Class Library structure	931	CRPEJob::GetExportOptions method, NewEra Class Library	976
CRPEExportOptions::CRPEExportOptions constructor, CRPE Class Library	932	CRPEJob::GetFormula method, CRPE Class Library	858
CRPEExportOptions::CRPEExportOptions constructor, NewEra Class Library	1043	CRPEJob::GetFormula method, NewEra Class Library	976
CRPEGraphDataInfo class, NewEra Class Library	1047	CRPEJob::GetGraphData method, CRPE Class Library	858
CRPEGraphDataInfo, CRPE Class Library structure	935	CRPEJob::GetGraphData method, NewEra Class Library	977
CRPEGraphDataInfo::CRPEGraphDataInfo constructor, CRPE Class Library	936	CRPEJob::GetGraphOptions method, CRPE Class Library	860
		CRPEJob::GetGraphOptions method, NewEra Class Library	978

CRPEJob::GetGraphText method, CRPE Class Library	861
CRPEJob::GetGraphText method, NewEra Class Library	980
CRPEJob::GetGraphType method, CRPE Class Library	862
CRPEJob::GetGraphType method, NewEra Class Library	981
CRPEJob::GetGroupCondition method, CRPE Class Library.....	864
CRPEJob::GetGroupCondition method, NewEra Class Library.....	983
CRPEJob::GetGroupSelection- Formula method, CRPE Class Library.....	868
CRPEJob::GetGroupSelection- Formula method, NewEra Class Library.....	986
CRPEJob::GetJobHandle method, CRPE Class Library	868
CRPEJob::GetJobHandle method, NewEra Class Library	987
CRPEJob::GetJobStatus method, CRPE Class Library	869
CRPEJob::GetJobStatus method, NewEra Class Library	987
CRPEJob::GetLineHeight method, NewEra Class Library	988
CRPEJob::GetMargins method, CRPE Class Library	870
CRPEJob::GetMargins method, NewEra Class Library	989
CRPEJob::GetMinimumSection- Height method, CRPE Class Library.....	870
CRPEJob::GetMinimumSection- Height method, NewEra Class Library.....	990
CRPEJob::GetNDetailCopies method, CRPE Class Library.....	872
CRPEJob::GetNDetailCopies method, NewEra Class Library.....	991
CRPEJob::GetNFormulas method, CRPE Class Library	872
CRPEJob::GetNFormulas method, NewEra Class Library	992
CRPEJob::GetNGroups method, CRPE Class Library	873
CRPEJob::GetNGroups method, NewEra Class Library	992
CRPEJob::GetNGroupSortFields method, CRPE Class Library	873
CRPEJob::GetNGroupSortFields method, NewEra Class Library	993
CRPEJob::GetNLinesInSection method, NewEra Class Library	993
CRPEJob::GetNPages method, CRPE Class Library.....	874
CRPEJob::GetNParameterFields method, CRPE Class Library	874
CRPEJob::GetNParams method, CRPE Class Library.....	875
CRPEJob::GetNParams method, NewEra Class Library.....	994
CRPEJob::GetNSections method, CRPE Class Library.....	875
CRPEJob::GetNSortFields method, CRPE Class Library.....	876
CRPEJob::GetNSortFields method, NewEra Class Library	995
CRPEJob::GetNSubreportsInSection method, CRPE Class Library	876
CRPEJob::GetNTables method, CRPE Class Library.....	877
CRPEJob::GetNTables method, NewEra Class Library	995
CRPEJob::GetNthFormula method, CRPE Class Library.....	877
CRPEJob::GetNthFormula method, NewEra Class Library	996
CRPEJob::GetNthGroupSortField method, CRPE Class Library	878
CRPEJob::GetNthGroupSortField method, NewEra Class Library	996
CRPEJob::GetNthParam method, CRPE Class Library.....	879
CRPEJob::GetNthParam method, NewEra Class Library	997
CRPEJob::GetNthParameterField method, CRPE Class Library	880
CRPEJob::GetNthParamInfo method, CRPE Class Library	880
CRPEJob::GetNthSortField method, CRPE Class Library.....	881
CRPEJob::GetNthSortField method, NewEra Class Library	998
CRPEJob::GetNthSubreportIn- Section method, CRPE Class Library.....	882
CRPEJob::GetNthTableLocation method, CRPE Class Library.....	883
CRPEJob::GetNthTableLocation method, NewEra Class Library.....	999
CRPEJob::GetNthTableLogOnInfo method, CRPE Class Library	883
CRPEJob::GetNthTableLogOnInfo method, NewEra Class Library	1000
CRPEJob::GetNthTableSessionInfo method, CRPE Class Library	884
CRPEJob::GetNthTableSessionInfo method, NewEra Class Library	1000
CRPEJob::GetNthTableType method, CRPE Class Library	885
CRPEJob::GetNthTableType method, NewEra Class Library	1001
CRPEJob::GetPrintDate method, CRPE Class Library	885
CRPEJob::GetPrintDate method, NewEra Class Library	1002
CRPEJob::GetPrintOptions method, CRPE Class Library	886
CRPEJob::GetPrintOptions method, NewEra Class Library	1002
CRPEJob::GetReportTitle method, CRPE Class Library	887
CRPEJob::GetReportTitle method, NewEra Class Library	1003
CRPEJob::GetSectionCode method, CRPE Class Library	887
CRPEJob::GetSectionFormat method, CRPE Class Library	888
CRPEJob::GetSectionFormat method, NewEra Class Library	1003, 1005
CRPEJob::GetSelectedPrinter method, CRPE Class Library	889
CRPEJob::GetSelectedPrinter method, NewEra Class Library	1006
CRPEJob::GetSelectionFormula method, CRPE Class Library	890

CRPEJob::GetSelectionFormula method, NewEra Class Library	1007	CRPEJob::SetFont method, CRPE Class Library CRPEJob::SetFont method, NewEra Class Library	899 1012
CRPEJob::GetSQLQuery method, CRPE Class Library	891	CRPEJob::SetFormula method, CRPE Class Library.....	903
CRPEJob::GetSQLQuery method, NewEra Class Library	1007	CRPEJob::SetFormula method, NewEra Class Library..... CRPEJob::SetGraphData method, CRPE Class Library.....	1016 903
CRPEJob::GetSubreportInfo method, CRPE Class Library	891	CRPEJob::SetGraphData method, NewEra Class Library..... CRPEJob::SetGraphOptions method, NewEra Class Library	1016 1018
CRPEJob::GetWindowHandle method, CRPE Class Library	892	CRPEJob::SetGraphText method, CRPE Class Library..... CRPEJob::SetGraphText method, NewEra Class Library.....	906 1019
CRPEJob::HasSavedData method, CRPE Class Library	892	CRPEJob::SetGraphType method, NewEra Class Library..... CRPEJob::SetGroupCondition method, CRPE Class Library	1020 909
CRPEJob::IsJobFinished method, CRPE Class Library	893	CRPEJob::SetGroupCondition method, NewEra Class Library	1022
CRPEJob::IsJobFinished method, NewEra Class Library	1008	CRPEJob::SetGroupSelection- Formula method, CRPE Class Library	912
CRPEJob::NextWindow- Magnification method, NewEra Class Library	1008	CRPEJob::SetGroupSelection- Formula method, NewEra Class Library	1025
CRPEJob::NextWindow- Magnification method, CRPE Class Library	894	CRPEJob::SetLineHeight method, NewEra Class Library..... CRPEJob::SetMargins method, CRPE Class Library.....	1025 912
CRPEJob::OpenSubreportJob method, CRPE Class Library	894	CRPEJob::SetMargins method, NewEra Class Library..... CRPEJob::SetMinimumSection- Height method, CRPE Class Library	1026 913
CRPEJob::OutputToPrinter method, CRPE Class Library	895	CRPEJob::SetMinimumSection- Height method, NewEra Class Library	1027
CRPEJob::OutputToPrinter method, NewEra Class Library	1009	CRPEJob::SetNDetailCopies method, CRPE Class Library	914
CRPEJob::OutputToWindow method, CRPE Class Library	895	CRPEJob::SetNDetailCopies method, NewEra Class Library	1028
CRPEJob::OutputToWindow method, NewEra Class Library	1009	CRPEJob::SetNGroupSortField method, CRPE Class Library	915
CRPEJob::PrintControlsShowing method, CRPE Class Library	897	CRPEJob::SetNGroupSortField method, NewEra Class Library	1029
CRPEJob::PrintControlsShowing method, NewEra Class Library	1010	CRPEJob::SetNParam method, CRPE Class Library	916
CRPEJob::PrintWindow method, CRPE Class Library	897	CRPEJob::SetNParam method, NewEra Class Library	1030
CRPEJob::PrintWindow method, NewEra Class Library	1011	CRPEJob::SetNTableLocation method, CRPE Class Library	917
CRPEJob::SelectPrinter method, CRPE Class Library	898	CRPEJob::SetNTableLogOnInfo method, CRPE Class Library	919
CRPEJob::SelectPrinter method, NewEra Class Library	1011	CRPEJob::SetNTableLogOnInfo method, NewEra Class Library	1032
CRPEJob::SetDialogParentWindow method, CRPE Class Library	899	CRPEJob::SetNTableSessionInfo method, CRPE Class Library	920
		CRPEJob::SetNTableSessionInfo method, NewEra Class Library	1033
		CRPEJob::SetPrintDate method, CRPE Class Library	921
		CRPEJob::SetPrintDate method, NewEra Class Library	1034
		CRPEJob::SetPrintOptions method, CRPE Class Library	922
		CRPEJob::SetPrintOptions method, NewEra Class Library	1035
		CRPEJob::SetReportTitle method, CRPE Class Library	922
		CRPEJob::SetReportTitle method, NewEra Class Library	1035
		CRPEJob::SetSectionFormat method, CRPE Class Library	923
		CRPEJob::SetSectionFormat method, NewEra Class Library	1036
		CRPEJob::SetSelectionFormula method, CRPE Class Library	924
		CRPEJob::SetSelectionFormula method, NewEra Class Library	1037
		CRPEJob::SetSQLQuery method, CRPE Class Library	925

CRPEJob::SetTableLocation method, NewEra Class Library.....	1032	CRPEParameterInfo, CRPE Class Library structure	947
CRPEJob::ShowFirstPage method, CRPE Class Library	925	CRPEParameterInfo:: CRPEParameterInfo constructor, CRPE Class Library	947
CRPEJob::ShowFirstPage method, NewEra Class Library	1038	CRPEPrintOptions class, NewEra Class Library.....	1055
CRPEJob::ShowLastPage method, CRPE Class Library	926	CRPEPrintOptions, CRPE Class Library structure	948
CRPEJob::ShowLastPage method, NewEra Class Library	1039	CRPEPrintOptions:: CRPEPrintOptions constructor, CRPE Class Library	948
CRPEJob::ShowNextPage method, CRPE Class Library	926	CRPEPrintOptions:: CRPEPrintOptions constructor, NewEra Class Library	1055
CRPEJob::ShowNextPage method, NewEra Class Library	1039	CRPESectionOptions class, NewEra Class Library.....	1056
CRPEJob::ShowPreviousPage method, CRPE Class Library.....	927	CRPESectionOptions, CRPE Class Library structure	949
CRPEJob::ShowPreviousPage method, NewEra Class Library.....	1040	CRPESectionOptions::CRPESection Options constructor, CRPE Class Library.....	951
CRPEJob::ShowPrintControls method, CRPE Class Library.....	928	CRPESessionOptions::CRPESession Options constructor, NewEra Class Library.....	1057
CRPEJob::ShowPrintControls method, NewEra Class Library.....	1040	CRPESessionInfo class, NewEra Class Library.....	1059
CRPEJob::Start method, CRPE Class Library.....	928	CRPESessionInfo, CRPE Class Library structure	952
CRPEJob::StartJob method, NewEra Class Library.....	1041	CRPESessionInfo::CRPESessionInfo constructor, CRPE Class Library.....	953
CRPEJob::TestNthTable- Connectivity method, CRPE Class Library.....	929	CRPESessionInfo::CRPESessionInfo constructor, NewEra Class Library.....	1059
CRPEJob::TestNthTable- Connectivity method, NewEra Class Library.....	1041	CRPESubreportInfo, CRPE Class Library structure	954
CRPEJob::ZoomPreviewWindow method, CRPE Class Library.....	929	CRPESubreportInfo::CRPESubreport Info constructor, CRPE Class Library.....	954
CRPEJob::ZoomPreviewWindow method, NewEra Class Library.....	1042	CRPETableLocation class, NewEra Class Library.....	1060
CRPEJob::public CObject class, CRPE Class Library	849	CRPETableLocation, CRPE Class Library structure	954
CRPEJob::SetSQLQuery method, NewEra Class Library	1038	CRPETableLocation:: CRPETableLocation constructor, CRPE Class Library.....	955
CRPEJobInfo class, NewEra Class Library.....	1052	CRPETableLocation::: CRPETableLocation constructor, NewEra Class Library.....	1060
CRPEJobInfo, CRPE Class structure	941	CRPETableType class, NewEra Class Library.....	1061
CRPEJobInfo::CRPEJobInfo constructor, CRPE Class Library.....	930	CRPETableType, CRPE Class Library structure	955

<p>CRPETableType::CRPETableType constructor, CRPE Class Library 956</p> <p>CRPETableType::CRPETableType constructor, NewEra Class Library 1061</p> <p>Crystal ActiveX Control 82</p> <p>Crystal ActiveX Control Methods 594</p> <p>Crystal Report Engine distributing applications 76 introduction to 44 structures 66 using 46 variable length strings 63</p> <p>Crystal Report Engine API 48 using in Visual Basic 78</p> <p>Crystal Report Engine Automation Server 26</p> <p>Crystal Reports Engine Class Library Reference 839</p> <p>Crystal Smart Viewer customizing 29 overview 32</p> <p>Crystal Smart Viewer/ActiveX 34</p> <p>Crystal Smart Viewer/HTML 39</p> <p>Crystal Smart Viewer/java 33</p> <p>Crystal Smart Viewers printing from 32</p> <p>Crystal VCL adding to project 108 changing properties 110 using 109</p> <p>Crystal VCL Overview 739</p> <p>Crystal Web Report Server administration 10 overview 10 parameters 13</p> <p>customizing the Crystal Smart Viewer 29</p> <p>custom-print link 53</p>	<p>Database Parameter Object, REOL 627</p> <p>DatabaseParameters Collection, REOL 629</p> <p>databases ODBC 18 secured 18 SQL 18</p> <p>DatabaseTable Object GetPrivateData method, REOL 632</p> <p>SetLogOnInfo method, REOL 632</p> <p>SetPrivateData method, REOL 633</p> <p>SetSessionInfo method, REOL 633</p> <p>TestConnectivity method, REOL 634</p> <p>DatabaseTable Object, REOL 630</p> <p>DatabaseTables Collection, REOL 635</p> <p>DataFiles property 500</p> <p>DataFiles property, VCL 743</p> <p>DataFilesLocation property, VCL 744</p> <p>DataSource property 502</p> <p>dBASE syntax PECancelPrintJob 122</p> <p>PECancCloseEngine 123</p> <p>PECheckFormula 124</p> <p>PECheckGroupSelection- Formula 126</p> <p>PECheckSelection- Formula 127</p> <p>PECloseEngine 128</p> <p>PEClosePrintJob 130</p> <p>PECloseSubreport 131</p> <p>PECloseWindow 133</p> <p>PEDeleteNthGroupSort- Field 138</p> <p>PEDeleteNthSortField 139</p> <p>PEDiscardSavedData 141</p> <p>PEExportPrintWindow 145</p> <p>PEGetErrorCode 153</p> <p>PEGetNFormulas 178</p> <p>PEGetNGroups 180</p> <p>PEGetNGroupSortFields 181</p> <p>PEGetNParams 185</p> <p>PEGetNSortFields 188</p> <p>PEGetNTables 191</p> <p>PEGetVersion 232</p> <p>PEGetWindowHandle 233</p> <p>PEIsPrintJobFinished 240</p> <p>PENextPrintWindow- Magnification 247</p> <p>PEOpenEngine 248</p> <p>PEOpenPrintJob 249</p> <p>PEOutputToPrinter 256</p> <p>PEOutputToWindow 261</p> <p>PEPrintReport 265</p> <p>PEPrintWindow 266</p> <p>PESetFont 287</p> <p>PESetFont 289</p> <p>PESetGroupCondition 301</p> <p>PESetGroupSelection- Formula 305</p> <p>PESetMargins 307</p> <p>PESetMinimumSection- Height 309</p> <p>PESetNDetailCopies 310</p> <p>PESetNthGroupSortField 313</p> <p>PESetNthParam 315</p> <p>PESetNthSortField 318</p> <p>PESetNthTableLocation 320</p> <p>PESetPrintDate 326</p> <p>PESetReportTitle 331</p> <p>PESetSelectionFormula 337</p> <p>PESetSQLQuery 339</p> <p>PEShowPrintControls 347</p> <p>PEStartPrintJob 349</p> <p>PETestNthTable- Connectivity 351</p> <p>PEZoomPreview- Window 352</p>
<p>DCU adding to project 108 changing properties 110 using 109</p> <p>DeactivatePrintWindow event, REOL 727</p> <p>Decoding 118</p> <p>Delete method, REOL 708</p> <p>Delphi adding VCL to project 108 changing VCL properties ... 110 using the Crystal VCL 109</p> <p>Delphi syntax DEVMODE 371</p> <p>PECancelPrintJob 122</p> <p>PECanCloseEngine 123</p> <p>PECheckFormula 124</p> <p>PECheckGroupSelection- Formula 125</p> <p>PECheckSelection- Formula 127</p> <p>PECloseButtonClickedEvent- Info 392</p> <p>PECloseEngine 128</p> <p>PEClosePrintJob 130</p>	

PECloseSubreport	131	PEReportSummaryInfo	441
PECloseWindow	132	PESearchButtonClicked-EventInfo.....	444
PEConvertPFIinfoVInfo....	134	PESectionOptions	447
PEConvertVInfotoPFIInfo....	136	PESelectPrinter.....	268
PEDeleteNthGroupSort-Field.....	137	PESessionInfo.....	449
PEDeleteNthSortField	139	PESetAreaFormat	270
PEDiscardSavedData	141	PESetAreaFormat-Formula	273
PEDrillOnDetailEvent-Info	393	PESetDialogParent-Window.....	275
PEDrillOnGroupEvent-Info	396	PESetEventCallback	281
PEEnableEvent	142	PESetFont.....	286
PEEnableEventInfo	399	PESetFormula.....	289
PEEnableProgressDialog	143	PESetGraphData	290
PEExportOptions	406	PESetGraphOptions	292
PEExportPrintWindow.....	145	PESetGraphText	294
PEExportTo	147	PESetGraphType	296
PEFieldValueInfo	408	PESetGroupCondition	301
PEGeneralPrintWindow-EventInfo	410	PESetGroupOptions	303
PEGetAreaFormat	148	PESetGroupSelection-Formula	305
PEGetAreaFormat-Formula	150	PESetMargins	307
PEGetEnableEventInfo.....	152	PESetMinimumSection-Height.....	309
PEGetErrorCode	153	PESetNDetailCopies.....	310
PEGetErrorText	154	PESetNthGroupSortField	312
PEGetExportOptions	156	PESetNthParam.....	314
PEGetFormula.....	157	PESetNthParameterField	316
PEGraphData	159	PESetNthSortField	318
PEGraphOptions	160	PESetNthTableLogOn-Info	322
PEGraphText	162	PESetNthTableSession-Info	323
PEGraphType	164	PESetPrintDate	325
PEGroupCondition	166	PESetPrintOptions	327
PEGroupOptions	168	PESetReportSummaryInfo	329
PEGroupSelection-Formula	169	PESetReportTitle	330
PEGetHandleString	171	PESetSectionFormat	332
PEGetJobStatus	173	PESetSectionFormat-Formula	335
PEGetMargins	174	PESetSelectionFormula	337
PEGetMinimumSection-Height	176	PESetSQLQuery	339
PEGetNDetailCopies	177	PESetTrackCursorInfo	340
PEGetNFormulas	178	PESetWindowOptions	342
PEGetNGroups	179	PEShowGroupEventInfo	451
PEGetNGroupSortFields....	181	PEShowNthPage	346
PEGetNPages.....	182	PEStartEventInfo	453
PEGetNParameterFields	184	PEStartPrintJob	349
PEGetNParams	185	PEStopEventInfo	455
PEGetNSections	186	PESubreportInfo	456
PEGetNSortFields.....	188	PETableLocation	458
PEGetNSubreportsIn-Section	189	PETableType	461
PEGetNTables.....	190	PETestNthTable-Connectivity.....	351
PEGetNthFormula	192	PETrackCursorInfo	465
PEGetNthGroupSortField ...	195		

PEValueInfo 469 PEWindowOptions 472 PEZoomLevelChanging- EventInfo 475 PEZoomPreviewWindow 352 ShowPrintControls 347 UXDDiskOptions 476 UXDMAPIOptions 477 UXDPostFolderOptions 480 UXDSMIOptions 479 UXDVIMOptions 482 UXFCharSeparated- Options 483 UXFCommaTabSeparated- Options 485 UXFDIOptions 486 UXFHMTL3Options 487 UXFODBCOptions 488 UXFPaginatedText- Options 490 UXFRecordStyleOptions 491 Designing UFL Functions 1066 designing web activity reports 20 Design-time ActiveX Control 26 Destination property 502 Destination property, VCL 744 DetailCopies property 503 DetailCopies property, VCL 745 developers what you should know 45 development Report Engine API 48 DEVMODE C syntax 353 Delphi syntax 371 VB syntax 370 DEVMODE structure 353 DialogParentHandle property 504 DialogParentHandle property, VCL 746 DiscardSavedData method, REOL 690 DiscardSavedData property 505 DiscardSavedData property, VCL 747 distributing Report Engine applications 76 drilling down on data 12 DrillOnDetail event, REOL 727 DrillOnGraph method, REOL 668 DrillOnGroup event, REOL 728	EMailCCList property, VCL 747 EMailMessage property 507 EMailMessage property, VCL 748 EMailSubject property 508 EMailSubject property, VCL 749 EMailToList property 508 EMailToList property, VCL 749 EMailVIMBCCLList property 509 EMailVIMBCCLList property, VCL 750 Encoding 116 Error Codes, REOL 736 Error Messages method 607 Error Table 1075 errors, preview window handling 71 EventInfo Object, REOL 636 ExchangeFolder property 510 ExchangeFolder property, VCL 751 ExchangePassword 511 ExchangePassword property, VCL 752 ExchangeProfile 511 ExchangeProfile property, VCL 753 Execute property, VCL 753 Export method, REOL 691, 721 ExportButtonClicked event, REOL 729 ExportOptions Object PromptForExportOptions method, REOL 641 Reset method, REOL 641 ExportOptions Object, REOL 637 ExportPrintWindow method, VCL 826	FormulaFieldDefinition Object, REOL 646 FormulaFieldDefinitions Collection, REOL 649 Formulas 512 Formulas property, VCL 754 formulas, selection changing in web reports 17 Function Data 1067 Function Definition 1071 Function Definition Table 1072 Function definition table example 1073 Function Examples Table 1074 Function Templates Table 1073 functions PECancelPrintJob 121 PECancCloseEngine 122 PECheckFormula 123 PECheckGroupSelection- Formula 124 PECheckSelection- Formula 126 PECloseEngine 127 PEClosePrintJob 129 PECloseSubreport 130 PECloseWindow 131 PEConvertPFIinfoVInfo 133 PEConvertVInfoPFIinfo 135 PEDeleteNthGroupSort- Field 136 PEDeleteNthSortField 138 PEDiscardSavedData 140 PEEnableEvent 141 PEEnableProgressDialog 142 PEExportPrintWindow 144 PEExportTo 146 PEGGetAreaFormat 147 PEGGetAreaFormat- Formula 148 PEGGetEnableEventInfo 151 PEGGetErrorCode 152 PEGGetErrorText 153 PEGGetExportOptions 155 PEGGetFormula 156 PEGGetGraphData 158 PEGGetGraphOptions 159 PEGGetGraphText 161 PEGGetGraphType 162 PEGGetGroupCondition 164 PEGGetGroupOptions 167 PEGGetGroupSelection- Formula 168 PEGGetHandleString 170 PEGGetJobStatus 171 PEGGetMargins 173
---	--	--

E

editing Active Server Pages 29
EMailCCList property 506

PEGetMinimumSection-Height.....	175
PEGetNDetailCopies.....	176
PEGetNFormulas	177
PEGetNGroups	179
PEGetNGroupSortFields....	180
PEGetNPages.....	181
PEGetNParameterFields	183
PEGetNParams	184
PEGetNSections	185
PEGetNSortFields.....	187
PEGetNSubreportsIn-Section.....	188
PEGetNTables.....	189
PEGetNth Formula	191
PEGetNthGroupSortField ...	193
PEGetNth Param	195
PEGetNth ParameterField ...	197
PEGetNthParamInfo	199
PEGetNthSortField.....	200
PEGetNthSubreportIn-Section.....	202
PEGetNthTableLocation....	203
PEGetNthTableLogOn-Info	205
PEGetNthTableSession-Info	206
PEGetNthTableType	207
PEGetPrintDate	209
PEGetPrintOptions.....	211
PEGetReportTitle	212, 214
PEGetSectionCode.....	215
PEGetSectionFormat	217
PEGetSectionFormat-Formula	219
PEGetSelectedPrinter	222
PEGetSelectionFormula	224
PEGetSQLQuery	226
PEGetSubreportInfo	228
PEGetTrackCursorInfo.....	229
PEGetVersion	231
PEGetWindowHandle.....	232
PEGetWindowOptions.....	234
PEHasSavedData	237
PEIsPrintJobFinished	238
PELogOffServer.....	240
PELogOnServer.....	241
PELogOnSqlServerWith-PrivateInfo.....	243
PENextPrintWindow-Magnification	245
PEOpenEngine	247
PEOpenPrintJob	248
PEOpenSubreport	250
PEOutputToFile	251
PEOutputToPrinter.....	254

PEOutputToWindow	257
PEPrintControlsShowing	261
PEPrintReport.....	263
PEPrintWindow.....	265
PESelectPrinter	267
PESetAreaFormat	269
PESetAreaFormat-Formula	271
PESetDialogParent-Window	274
PESetEventCallback	275
PESetFont	282
PESetFormula	287
PESetGraphData	289
PESetGraphOptions	291
PESetGraphText	292
PESetGraphType	294
PESetGroupCondition	296
PESetGroupOptions	302
PESetGroupSelection-Formula	303
PESetMargins	305
PESetMinimumSection-Height	307
PESetNDetailCopies	309
PESetNthGroupSortField ...	311
PESetNthParam	313
PESetNthParameterField ...	315
PESetNthSortField	316
PESetNthTableLocation	319
PESetNthTableLogOn-Info	320
PESetNthTableSession-Info	322
PESetPrintDate	324
PESetPrintOptions	326
PESetReportSummary-Info	327
PESetReportTitle	329
PESetSectionFormat	331
PESetSectionFormat-Formula	333
PESetSelectionFormula	336
PESetSQLQuery.....	337
PESetTrackCursorInfo	339
PESetWindowOptions	341
PEShowFirstPage	342
PEShowLastPage	342
PEShowNextPage	342
PEShowNthPage	345
PEShowPreviousPage	342
PEShowPrintControls.....	346
PEStartPrintJob	348
PETestNthTableTable-Connectivity	349
PEZoomPreviewWindow...	351

G

GetNSubreports method	594
GetNthSubreportName method	595
GetPageNumberForGroup method, REOL	669
GetPrivateData method, REOL	632
GlobalOptions Object, REOL	650
GraphData	513
GraphData property, VCL	755
GraphObject Object, REOL	651
GraphOptions	515
GraphOptions property, VCL	757
GraphText	517
GraphText property, VCL	759
GraphType	518
GraphType property, VCL	760
GroupAreaOptions Object, REOL	656
GroupCondition	520
GroupCondition property, VCL	761
GroupNameFieldDefinition Object, REOL	658
GroupSelectionFormula	522
GroupSelectionFormula property, VCL	763
GroupSortFields	523
GroupSortFields property, VCL	764
GroupTreeButtonClicked event, REOL	730

H

handling preview window errors	71
Helper Modules	1069
how to design web activity reports	20
how to request web activity reports from your browser	21
HTML reports limitations of	39
HTML, Crystal Smart Viewer	39

I

Implementing InitJob and TermJob	1085
InitForJob Function	1076

J

Java, Crystal Smart Viewer	33
----------------------------------	----

K	
Known problems with earlier versions of UFJob.C.....	1085
L	
LastErrorNumber	524
LastErrorNumber property, VCL.....	765
LastErrorString	525
LastErrorString property, VCL	766
LastPageButtonClicked event, REOL	730
limitations of HTML reports.....	39
LineObject Object, REOL.....	660
link	
custom-print	53
print-only	50
linking to web reports	12
LogoffServer method	595
LogOffServer method, REOL	613
LogoffServer method, VCL.....	827
LogOnInfo.....	526
LogOnInfo property, VCL.....	767
LogonServer method	596
LogOnServer method, REOL	614
LogonServer method, VCL.....	828
M	
MarginBottom	528
MarginBottom property, VCL	768
MarginLeft.....	529
MarginLeft property, VCL	769
MarginRight	529
MarginRight property, VCL.....	770
MarginTop	530
MarginTop property, VCL.....	770
MDIChild property, VCL	771
Microsoft AuthentiCode	35
mime types	11
Module Definition (.def)	
File.....	1082
monitoring web activity with reports.....	20
N	
NewEra Class Library	
CRPEExportOptions	
class.....	1043
CRPEExportOptions::	
CRPEExportOptions	
constructor.....	1043
CRPEGraphDataInfo	
class.....	1047
CRPEGraphDataInfo::	
CRPEGraphDataInfo	
constructor	1047
CRPEGraphOptions	
class	1049
CRPEGraphOptions::	
CRPEGraphOptions	
constructor	1049
CRPEGraphTextInfo	
class	1050
CRPEGraphTextInfo::	
CRPEGraphTextInfo	
constructor	1051
CRPEJob class	969
CRPEJob::Cancel method	969
CRPEJob::CheckFormula	
method	970
CRPEJob::	
CheckGroupSelectionFormula	
method	970
CRPEJob::	
CheckSelectionFormula	
method	971
CRPEJob::Close method	971
CRPEJob::CloseWindow	
method	971
CRPEJob::CRPEJob	
constructor	972
CRPEJob::	
DeleteNthGroupSortField	
method	972
CRPEJob::	
DeleteNthSortField	
method	973
CRPEJob::ExportPrintWindow	
method	974
CRPEJob::ExportTo	
method	974
CRPEJob::GetErrorCode	
method	975
CRPEJob::GetErrorText	
method	975
CRPEJob::GetExportOptions	
method	976
CRPEJob::GetFormula	
method	976
CRPEJob::GetGraph Data	
method	977
CRPEJob::GetGraph Options	
method	978
CRPEJob::GetGraph Text	
method	980
CRPEJob::GetGraph Type	
method	981
CRPEJob::GetGroupCondition	
method	983
CRPEJob::GetGroupSelectionFormula	
method	986
CRPEJob::GetJobHandle	
method	987
CRPEJob::GetJobStatus	
method	987
CRPEJob::GetLineHeight	
method	988
CRPEJob::GetMargins	
method	989
CRPEJob::GetMinimumSectionHeight	
method	990
CRPEJob::GetNDetailCopies	
method	991
CRPEJob::GetNFormulas	
method	992
CRPEJob::GetNGroups	
method	992
CRPEJob::GetNGroupSortFields	
method	993
CRPEJob::GetNLinesInSection	
method	993
CRPEJob::GetNParams	
method	994
CRPEJob::GetNSortFields	
method	995
CRPEJob::GetNTables	
method	995
CRPEJob::GetNthFormula	
method	996
CRPEJob::GetNthGroupSortField	
method	996
CRPEJob::GetNthParam	
method	997
CRPEJob::GetNthSortField	
method	998
CRPEJob::GetNthTableLocation	
method	999
CRPEJob::GetNthTableLogOnInfo	
method	1000
CRPEJob::GetNthTableSessionInfo	
method	1000
CRPEJob::GetNthTableType	
method	1001
CRPEJob::GetPrintDate	
method	1002
CRPEJob::GetPrintOptions	
method	1002
CRPEJob::GetReportTitle	
method	1003
CRPEJob::GetSectionFormat	
method	1003, 1005
CRPEJob::GetSelectedPrinter	
method	1006
CRPEJob::GetSelectionFormula	
method	1007

CRPEJob::GetSQLQuery method	1007	CRPEJob::SetReportTitle method	1035	CRPEngine::OpenJob method	967
CRPEJob::IsJobFinished method	1008	CRPEJob::SetSectionFormat method	1036	CRPEngine::PrintReport method	968
CRPEJob::NextWindow- Magnification method	1008	CRPEJob::SetSelectionFormula method	1037	CRPEPrintOptions class ...	1055
CRPEJob::OutputToPrinter method	1009	CRPEJob::SetSQLQuery method	1038	CRPEPrintOptions:: CRPEPrintOptions constructor.....	1055
CRPEJob::OutputToWindow method	1009	CRPEJob::ShowFirstPage method	1038	CRPESectionOptions class.....	1056
CRPEJob::PrintControls- Showing method	1010	CRPEJob::ShowLastPage method	1039	CRPESectionOptions:: CRPESectionOptions constructor.....	1057
CRPEJob::PrintWindow method	1011	CRPEJob::ShowNextPage method	1039	CRPESessionInfo class....	1059
CRPEJob::SelectPrinter method	1011	CRPEJob::ShowPreviousPage method	1040	CRPESessionInfo:: CRPESessionInfo constructor.....	1059
CRPEJob::SetFont method	1012	CRPEJob::ShowPrintControls method	1040	CRPETableLocation class.....	1060
CRPEJob::SetFormula method	1016	CRPEJob::StartJob method	1041	CRPETableLocation:: CRPETableLocation constructor.....	1060
CRPEJob::SetGraph Data method	1016	CRPEJob::TestNthTable- Connectivity method	1041	CRPETableType class	1061
CRPEJob::SetGraphOptions method	1018	CRPEJob::ZoomPreview- Window method	1042	CRPETableType::CRPETable- Type constructor	1061
CRPEJob::SeGraph Text method	1019	CRPEJobInfo class	1052	NextMagnification method, REOL.....	721
CRPEJob::SetGraphType method	1020	CRPELogOnInfo class.....	1053	NextPageButtonClicked event, REOL.....	731
CRPEJob::SetGroupCondition method	1022	CRPELogOnInfo::CRPELog- OnInfo constructor	1054	NumberOfGroup property, VCL.....	772
CRPEJob::SetGroupSelection- Formula method.....	1025	CRPEngine class.....	960		
CRPEJob::SetLineHeight method	1025	CRPEngine::CanClose method	960		
CRPEJob::SetMargins method	1026	CRPEngine::Close method	960		
CRPEJob::SetMinimum- SectionHeight method....	1027	CRPEngine::CRPEngine constructor	961		
CRPEJob::SetNDetailCopies method	1028	CRPEngine::GetEngineStatus method	961		
CRPEJob::SetNthGroupSort- Field method	1029	CRPEngine::GetErrorCode method	962		
CRPEJob::SetNthParam method	1030	CRPEngine::GetErrorText method	962		
CRPEJob::SetNthSortField method	1031	CRPEngine::GetNPrintJobs method	963		
CRPEJob::SetNthTableLocation method	1032	CRPEngine::GetVersion method	963		
CRPEJob::SetNthTableLogOn- Info method	1032	CRPEngine::LogOffServer method	964		
CRPEJob::SetNthTableSession- Info method	1033	CRPEngine::LogOnServer method	964		
CRPEJob::SetPrintDate method	1034	CRPEngine::LogOnSQLServer- WithPrivateInfo method... <td>965</td> <td></td> <td></td>	965		
CRPEJob::SetPrintOptions method	1035	CRPEngine::Open method	967		

P	
Page Object	
RenderEPF method, REOL663	
RenderHTML method, REOL663	
Page Object, REOL662	
PageCount method597	
PageCount method, VCL829	
PageEngine Object	
CreatePageGenerator method, REOL665	
RenderTotalIlderETF method, REOL665	
RenderTotalIlderHTML method, REOL666	
PageEngine Object, REOL664	
PageFirst method598	
PageGenerator Object	
DrillOnGraph method, REOL668	
GetPageNumberForGroup method, REOL669	
SearchForText method, REOL669	
PageGenerator Object, REOL667	
PageLast method598	
PageNext method598	
PagePrevious method599	
Pages Collection, REOL670	
pages, active server	
editing29	
pages, web	
sample23	
PageSetup Object, REOL671	
PageShow method599	
PageZoom method600	
PageZoomNext method600	
parameter fields19	
ParameterFieldDefinition Object	
SetCurrentValue method, REOL677	
SetDefaultValue method, REOL678	
ParameterFieldDefinition Object, REOL673	
ParameterFieldDefinitions Collection, REOL679	
ParameterFields531	
parameters	
Crystal Web Report Server ...13	
ParamFields property, VCL772	
Password532	
Password property, VCL773	
PECancelPrintJob	
C syntax121	
dBASE syntax122	
Delphi syntax122	
VB syntax121	
PECancelCloseEngine	
C syntax122	
dBASE syntax123	
Delphi syntax123	
VB syntax122	
PECharSepFileOptions	
structure389	
PECheckFormula	
C syntax123	
dBASE syntax124	
Delphi syntax124	
VB syntax124	
PECheckGroupSelectionFormula	
C syntax124	
dBASE syntax126	
Delphi syntax125	
VB syntax125	
PECheckSelectionFormula	
C syntax126	
dBASE syntax127	
Delphi syntax127	
VB syntax127	
PECloseButtonClickedEventInfo	
Delphi syntax392	
VB syntax391	
PECloseButtonClickedEventInfo	
structure391	
C syntax391	
PECloseEngine	
C syntax127	
dBASE syntax128	
Delphi syntax128	
VB syntax128	
PEClosePrintJob	
C syntax129	
dBASE syntax130	
Delphi syntax130	
VB syntax129	
PECloseSubreport	
C syntax130	
dBASE syntax131	
Delphi syntax131	
VB syntax131	
PECloseWindow	
C syntax131	
dBASE syntax133	
Delphi syntax132	
VB syntax132	
PEConvertPFIinfoVInfo	
C syntax133	
Delphi syntax134	
VB syntax134	
PEConvertVInfoToPFIinfo	
C syntax135	
Delphi syntax136	
VB syntax135	
PEDeleteNthGroupSortField	
C syntax136	
dBASE syntax138	
Delphi syntax137	
VB syntax137	
PEDeleteNthSortField	
C syntax138	
dBASE syntax139	
Delphi syntax139	
VB syntax139	
PEDiscardSavedData	
C syntax140	
dBASE syntax141	
Delphi syntax141	
VB syntax140	
PEDrillOnDetailEventInfo	
C syntax392	
Delphi syntax393	
PEDrillOnDetailEventInfo	
structure392	
PEDrillOnGroupEventInfo	
C syntax394	
Delphi syntax396	
VB syntax395	
PEDrillOnGroupEventInfo	
structure394	
PEEnableEvent	
C syntax141	
Delphi syntax142	
VB syntax142	
PEEnableEventInfo	
C syntax397	
Delphi syntax399	
VB syntax398	
PEEnableEventInfo structure397	
PEEnableProgressDialog	
C syntax142	
Delphi syntax143	
VB syntax143	
PEExportOptions	
C syntax399	
Delphi syntax406	
VB syntax405	
PEExportOptions structure399	
PEExportPrintWindow	
C syntax144	
dBASE syntax145	
Delphi syntax145	
VB syntax145	
PEExportTo	
C syntax146	

Delphi syntax	147
VB syntax	146
PEFieldValuelnfo	
C syntax.....	407
Delphi syntax	408
VB syntax	408
PEFieldValuelnfo structure.....	407
PEGeneralPrintWindowEventlnfo	
C syntax.....	409
Delphi syntax	410
VB syntax	409
PEGeneralPrintWindowEventlnfo structure	409
PEGetAreaFormat	
C syntax.....	147
Delphi syntax	148
VB syntax	148
PEGetAreaFormatFormula	
C syntax.....	148
Delphi syntax	150
VB syntax	150
PEGetEnableEventlnfo	
C syntax.....	151
Delphi syntax	152
VB syntax	151
PEGetErrorCode	
C syntax.....	152
dBASE syntax.....	153
Delphi syntax	153
VB syntax	153
PEGetErrorText	
C syntax.....	153
Delphi syntax	154
VB syntax	154
PEGetExportOptions	
C syntax.....	155
Delphi syntax	156
VB syntax	155
PEGetFormula	
C syntax.....	156
Delphi syntax	157
VB syntax	157
PEGetGraphData	
C syntax.....	158
Delphi syntax	159
VB syntax	159
PEGetGraphOptions	
C syntax.....	159
Delphi syntax	160
VB syntax	160
PEGetGraphText	
C syntax.....	161
Delphi syntax	162
VB syntax	162
PEGetGraphType	
C syntax.....	162
Delphi syntax	164
VB syntax	163
PEGetGroupCondition	
C syntax	164
Delphi syntax	166
VB syntax	166
PEGetGroupOptions	
C syntax	167
Delphi syntax	168
VB syntax	167
PEGetGroupSelectionFormula	
C syntax	168
Delphi syntax	169
VB syntax	169
PEGetHandleString	
C syntax	170
Delphi syntax	171
VB syntax	171
PEGetJobStatus	
C syntax	171
Delphi syntax	173
VB syntax	172
PEGetMargins	
C syntax	173
Delphi syntax	174
VB syntax	174
PEGetMinimumSectionHeight	
C syntax	175
Delphi syntax	176
VB syntax	175
PEGetNDetailCopies	
C syntax	176
Delphi syntax	177
VB syntax	177
PEGetNFormulas	
C syntax	177
dBASE syntax	178
Delphi syntax	178
VB syntax	178
PEGetNGroups	
C syntax	179
dBASE syntax	180
Delphi syntax	179
VB syntax	179
PEGetNGroupSortFields	
C syntax	180
dBASE syntax	181
Delphi syntax	181
VB syntax	181
PEGetNPages	
C syntax	181
Delphi syntax	182
VB syntax	182
PEGetNParameterFields	
C syntax	183
Delphi syntax	184
VB syntax	183
PEGetNParams	
C syntax.....	184
dBASE syntax	185
Delphi syntax	185
VB syntax	185
PEGetNSections	
C syntax.....	185
Delphi syntax	186
VB syntax	186
PEGetNSortFields	
C syntax.....	187
dBASE syntax	188
Delphi syntax	188
VB syntax	187
PEGetNSubreportsInSection	
C syntax.....	188
Delphi syntax	189
VB syntax	189
PEGetNTables	
C syntax.....	189
dBASE syntax	191
Delphi syntax	190
VB syntax	190
PEGetNthFormula	
C syntax.....	191
Delphi syntax	192
VB syntax	192
PEGetNthGroupSortField	
C syntax.....	193
Delphi syntax	195
VB syntax	194
PEGetNthParam	
C syntax.....	195
Delphi syntax	197
VB syntax	196
PEGetNthParameterField	
C syntax.....	197
Delphi syntax	198
VB syntax	198
PEGetNthParamInfo	
C syntax.....	199
Delphi syntax	200
VB syntax	199
PEGetNthSortField	
C syntax.....	200
Delphi syntax	201
VB syntax	201
PEGetNthSubreportInSection	
C syntax.....	202
Delphi syntax	203
VB syntax	203
PEGetNthTableLocation	
C syntax.....	203
Delphi syntax	204
VB syntax	204

PEGetNthTableLogOnInfo	
C syntax	205
Delphi syntax	206
VB syntax	205
PEGetNthTableSessionInfo	
C syntax	206
Delphi syntax	207
VB syntax	207
PEGetNthTableType	
C syntax	207
Delphi syntax	209
VB syntax	208
PEGetPrintDate	
C syntax	209
Delphi syntax	210
VB syntax	210
PEGetPrintOptions	
C syntax	211
Delphi syntax	212
VB syntax	212
PEGetReportTitle	
C syntax	212, 214
Delphi syntax	214, 215
VB syntax	213, 215
PEGetSectionCode	
C syntax	215
Delphi syntax	216
VB syntax	216
PEGetSectionFormat	
C syntax	217
Delphi syntax	218
VB syntax	218
PEGetSectionFormatFormula	
C syntax	219
Delphi syntax	221
VB syntax	221
PEGetSelectedPrinter	
C syntax	222
Delphi syntax	224
VB syntax	223
PEGetSelectionFormula	
C syntax	224
Delphi syntax	225
VB syntax	225
PEGetSQLQuery	
C syntax	226
Delphi syntax	227
VB syntax	227
PEGetSubreportInfo	
C syntax	228
Delphi syntax	229
VB syntax	228
PEGetTrackCursorInfo	
C syntax	229
Delphi syntax	230
VB syntax	230
PEGetVersion	
C syntax	231
dBASE syntax	232
Delphi syntax	232
VB syntax	231
PEGetWindowHandle	
C syntax	232
dBASE syntax	233
Delphi syntax	233
VB syntax	233
PEGetWindowOptions	
C syntax	234
Delphi syntax	235
VB syntax	235
PEGraphDataInfo	
C syntax	410
Delphi syntax	412
VB syntax	412
PEGraphDataInfo structure	410
PEGraphOptions	
C syntax	413
Delphi syntax	415
VB syntax	414
PEGraphOptions structure	413
PEGraphTextInfo	
C syntax	415
Delphi syntax	417
VB syntax	416
PEGraphTextInfo structure	415
PEGroupOptions	
C syntax	418
Delphi syntax	420
VB syntax	420
PEGroupOptions structure	418
PEGroupTreeButtonClickedEvent- Info	
C syntax	421
Delphi syntax	422
VB syntax	422
PEGroupTreeButtonClickedEvent- Info structure	421
PEHasSavedData	
C syntax	237
Delphi syntax	238
VB syntax	238
PEIsPrintJobFinished	
C syntax	238
dBASE syntax	240
Delphi syntax	239
VB syntax	239
PEJobInfo	
C syntax	423
Delphi syntax	424
VB syntax	424
PEJobInfo structure	423
PELogOffServer	
C syntax	240
Delphi syntax	241
VB syntax	241
PELogOnInfo	
C syntax	425
Delphi syntax	427
VB syntax	426
PELogOnInfo structure	425
PELogOnServer	
C syntax	241
Delphi syntax	243
VB syntax	242
PELogOnSQLServerWithPrivateInfo	
C syntax	243
Delphi syntax	245
VB syntax	245
PENextPrintWindowMagnification	
C syntax	245
dBASE syntax	247
Delphi syntax	246
VB syntax	246
PEOpenEngine	
C syntax	247
dBASE syntax	248
Delphi syntax	247
VB syntax	247
PEOpenPrintJob	
C syntax	248
dBASE syntax	249
Delphi syntax	249
VB syntax	249
PEOpenSubreport	
C syntax	250
Delphi syntax	251
VB syntax	251
PEOutputToFile	
C syntax	251
PEOutputToPrinter	
C syntax	254
dBASE syntax	256
Delphi syntax	256
VB syntax	256
PEOutputToWindow	
C syntax	257
dBASE syntax	261
Delphi syntax	261
VB syntax	260
PEParameterFieldInfo	
C syntax	427
Delphi syntax	431
VB syntax	430
PEParameterFieldInfo structure	427
PEParameterInfo	
C syntax	432

Delphi syntax	434	PESetAreaFormat	C syntax	269	PESetMargins	C syntax.....	305
VB syntax	433	Delphi syntax.....	270	dBASE syntax.....	307		
PEParameterInfo structure.....	432	VB syntax.....	270	Delphi syntax	307		
PEPrintControlsShowing		PESetAreaFormatFormula	C syntax	271	VB syntax	306	
C syntax.....	261	Delphi syntax.....	273	PESetMinimumSectionHeight	C syntax.....	307	
Delphi syntax	262	VB syntax.....	273	dBASE syntax.....	309		
VB syntax	262	PESetDialogParentWindow	C syntax	274	Delphi syntax	309	
PEPrintFileOptions structure.....	434	Delphi syntax.....	275	VB syntax	308		
PEPrintOptions		VB syntax	274	PESetNDetailCopies	C syntax.....	309	
C syntax.....	435	Delphi syntax.....	275	dBASE syntax.....	310		
Delphi syntax	437	VB syntax	274	Delphi syntax	310		
VB syntax	437	PESetEventCallback	C syntax	275	VB syntax	310	
PEPrintOptions structure.....	435	Delphi syntax.....	281	PESetNthGroupSortField	C syntax.....	311	
PEPrintReport		VB syntax	281	dBASE syntax.....	313		
C syntax.....	263	PESetFont	C syntax	282	Delphi syntax	312	
dBASE syntax.....	265	dBASE syntax	287	VB syntax	312		
Delphi syntax	264	Delphi syntax	286	PESetNthParam	C syntax.....	313	
VB syntax	264	VB syntax	286	dBASE syntax.....	315		
PEPrintWindow		PESetFormula	C syntax	287	Delphi syntax	314	
C syntax.....	265	dBASE syntax	289	VB syntax	314		
dBASE syntax.....	266	Delphi syntax	289	PESetNthParameterField	C syntax.....	315	
Delphi syntax	266	VB syntax	288	Delphi syntax	316		
VB syntax	266	PESetGraphData	C syntax	289	VB syntax	316	
PEReadingRecordsEventInfo		Delphi syntax	290	PESetNthSortField	C syntax.....	316	
C syntax.....	438	VB syntax	290	dBASE syntax.....	318		
Delphi syntax	439	PESetGraphOptions	C syntax	291	Delphi syntax	318	
VB syntax	439	Delphi syntax	292	VB syntax	318		
PEReadingRecordsEventInfo		VB syntax	292	PESetNthTableLocation	C syntax.....	319	
structure	438	PESetGraphText	C syntax	292	dBASE syntax.....	320	
PREportSummaryInfo		Delphi syntax	294	VB syntax	319		
C syntax.....	440	PESetGraphType	C syntax	294	PESetNthTableLogOnInfo	C syntax.....	320
Delphi syntax	441	Delphi syntax	296	Delphi syntax	322		
VB syntax	441	VB syntax	296	PESetNthTableSessionInfo	C syntax.....	322	
PREportSummaryInfo		PESetGroupCondition	C syntax	296	Delphi syntax	323	
structure	440	dBASE syntax	301	VB syntax	323		
PESearchButtonClickedEventInfo		Delphi syntax	301	PESetPrintDate	C syntax.....	324	
C syntax.....	442	VB syntax	300	Delphi syntax	325		
Delphi syntax	444	PESetGroupOptions	C syntax	302	VB syntax	325	
VB syntax	443	Delphi syntax	303	PESetPrintDateInfo	dBASE syntax.....	326	
PESectionOptions		VB syntax	303	PESetPrintOptions	C syntax.....	326	
C syntax.....	444	PESetGroupSelectionFormula	C syntax	303	Delphi syntax	327	
Delphi syntax	447	dBASE syntax	305	VB syntax	327		
VB syntax	446	Delphi syntax.....	305				
PESectionOptions structure	444	VB syntax	304				
PESelectPrinter							
C syntax.....	267						
Delphi syntax	268						
VB syntax	268						
PESessionInfo							
C syntax.....	448						
Delphi syntax	449						
VB syntax	449						
PESessionInfo structure.....	448						

PESetReportSummaryInfo	
C syntax	327
Delphi syntax	329
VB syntax	328
PESetReportTitle	
C syntax	329
dBASE syntax.....	331
Delphi syntax	330
VB syntax.....	330
PESetSectionFormat	
C syntax	331
Delphi syntax	332
VB syntax.....	332
PESetSectionFormatFormula	
C syntax	333
Delphi syntax	335
VB syntax.....	335
PESetSelectionFormula	
C syntax	336
dBASE syntax.....	337
Delphi syntax	337
VB syntax	337
PESetSQLQuery	
C syntax	337
dBASE syntax.....	339
Delphi syntax	339
VB syntax	338
PESetTrackCursorInfo	
C syntax	339
Delphi syntax	340
VB syntax	340
PESetWindowOptions	
C syntax	341
Delphi syntax	342
VB syntax	341
PEShow Functions (dBASE for Windows Syntax)	344
PEShow Functions (Delphi Syntax)	344
PEShow Functions (VB Syntax)	343
PEShowFirstPage	
C syntax	342
PEShowGroupEventInfo	
C syntax	450
Delphi syntax	451
PEShowGroupEventInfo structure	450
PEShowLastPage	
C syntax	342
PEShowNextPage	
C syntax	342
PEShowNthPage	
C syntax	345
Delphi syntax	346
VB syntax	345
PEShowPreviousPage	
C syntax.....	342
PEShowPrintControls	
C syntax.....	346
dBASE syntax	347
Delphi syntax.....	347
VB syntax.....	347
PEStartEventInfo	
C syntax.....	451
Delphi syntax.....	453
VB syntax.....	452
PEStartEventInfo structure	451
PEStartPrintJob	
C syntax.....	348
dBASE syntax	349
Delphi syntax.....	349
VB syntax.....	349
PEStopEventInfo	
C syntax.....	453
Delphi syntax.....	455
VB syntax.....	454
PEStopEventInfo structure	453
PESubreportInfo	
C syntax.....	455
Delphi syntax.....	456
VB syntax.....	456
PESubreportInfo structure	455
PETableLocation	
C syntax.....	457
Delphi syntax.....	458
VB syntax.....	458
PETableLocation structure	457
PETableType	
C syntax.....	459
Delphi syntax.....	461
VB syntax.....	460
PETableType structure	459
PETestNthTableConnectivity	
C syntax.....	349
dBASE syntax	351
Delphi syntax.....	351
VB syntax.....	350
PETrackCursorInfo	
C syntax.....	461
Delphi syntax.....	465
VB syntax.....	464
PETrackCursorInfo structure	461
PEValueInfo	
C syntax.....	466
Delphi syntax.....	469
VB syntax.....	468
PEValueInfo structure	466
PEWindowOptions	
C syntax.....	470
Delphi syntax.....	472
VB syntax.....	471
PEWindowOptions structure	470
PEZoomLevelChangingEventInfo	
C syntax	473
Delphi syntax	475
VB syntax	474
PEZoomLevelChangingEventInfo structure	473
PEZoomPreviewWindow	
C syntax	351
dBASE syntax.....	352
Delphi syntax	352
VB syntax	352
Picture Function - a sample UFL function	1079
Preview method, REOL	691
preview window errors handling	71
PrevPageButtonClicked event, REOL	732
PrintButtonClicked event, REOL	732
PrintControls property, VCL	774
PrintDay	533
PrintDay property, VCL	774
PrintEnded property, VCL	775
PrinterCollation	534
PrinterCollation property, VCL	776
PrinterCopies	535
PrinterCopies property, VCL	777
PrinterDriver	536
PrinterDriver property, VCL	777
PrinterInfo Object, REOL	681
PrinterMode property, VCL	779
PrinterName	537
PrinterName property, VCL	780
PrinterPort	538
PrinterPort property, VCL	781
PrinterSelect method	601
PrinterStartPage	539
PrinterStartPage property, VCL	783
PrinterStopPage	540
PrinterStopPage property, VCL	783
PrintFileCharSepQuote	541
PrintFileCharSepQuote property, VCL	784
PrintFileCharSepSeparator	542
PrintFileCharSepSeparator property, VCL	785
PrintFileLinesPerPage	542
PrintFileLinesPerPage property, VCL	786
PrintFileName	543
PrintFileName property, VCL	786

PrintFileODBCPassword	544	REC Library	111
PrintFileODBCPassword property, VCL.....	788	structures	66
PrintFileODBCSource	545	variable length strings	63
PrintFileODBCSource property, VCL.....	789	RecordsPrinted property	554
PrintFileODBCTable	545	RecordsPrinted property, VCL	796
PrintFileODBCTable property, VCL.....	789	RecordsRead property	555
PrintFileODBCUser	546	RecordsRead property, VCL	797
PrintFileODBCUser property, VCL.....	790	RecordsSelected property	555
PrintFileType	547	RecordsSelected property, VCL	798
PrintFileType property, VCL	791	RefreshButtonClicked event, REOL	733
PrintFileUseRptDateFmt	549	refreshing web report data	17
PrintFileUseRptDateFmt property, VCL.....	793	RenderEPF method, REOL	663
PrintFileUseRptNumberFmt	550	RenderHTML method, REOL	663
PrintFileUseRptNumberFmt property, VCL	793	RenderTallerETF method, REOL	665
printing from the Crystal Smart Viewers	32	RenderTallerHTML method, REOL	666
PrintingStatus Object, REOL	681	REOL	
PrintMonth property	551	ActivatePrintWindow event	725
PrintMonth property, VCL	794	Add method	707
print-only link	50	AddGroup method	689
PrintOut method, REOL	692, 721	Application Object	611
PrintReport method	601	Area Object	615
PrintSetupButtonClicked event, REOL	733	AreaOptions Object	617
PrintWindow method, VCL	829	Areas Collection	619
PrintWindowOptions Object, REOL	683	BlobField Object	620
PrintYear	552	Box Object	621
PrintYear property, VCL	795	CancelButtonClicked event	725
procedures, stored SQL	19	CancelPrinting method	690
programming		CanClose method	612
Report Engine API	48	Check method	649
Programming the UFL	1069	ClearError method	613, 690
ProgressDialog	553	Close method	720, 724
ProgressDialog property, VCL	796	CloseButtonClicked event	726
PromptForExportOptions method, REOL	641	ClosePrintWindow event	726
properties		CreatePageGenerator method	665
changing for ActiveX Control	84	CrossTabObject Object	622
changing VCL	110	Database Object	623
Purpose of the function	1067	DatabaseFieldDefinition Object	624
R		DatabaseFieldDefinitions Collection	626
ReadingRecords event, REOL	694	DatabaseParameter Object	627
ReadRecords method, REOL	693	DatabaseParameters Collection	629
REAPI	48	DatabaseTables Collection	630
		DatabaseTables Collection	635
		DeactivatePrintWindow event	727
		Delete method	708
		DiscardSavedData method	690
		DrillOnDetail event	727
		DrillOnGraph method	668
		DrillOnGroup event	728
		Error Codes	736
		EventInfo Object	636
		Export method	691, 721
		ExportButtonClicked event	729
		ExportOptions Object	637
		FieldDefinitions Collection	642
		FieldObject Object	642
		FieldValue Object	644
		FirstPageButtonClicked event	729
		Font Object	644
		FormulaFieldDefinition Object	646
		FormulaFieldDefinitions Collection	649
		GetPageNumberForGroup method	669
		GetPrivateData method	632
		GlobalOptions Object	650
		GraphObject Object	651
		GroupAreaOptions Object	656
		GroupNameField Definition Object	658
		GroupTreeButtonClicked event	730
		LastPageButtonClicked event	730
		LineObject Object	660
		LogOffServer method	613
		LogOnServer method	614
		NextMagnification method	721
		NextPageButtonClicked event	731
		OLEObject Object	661
		OpenReport method	614
		OpenSubreport method	691
		Page Object	662
		PageEngine Object	664
		PageGenerator Object	667
		Pages Collection	670
		PageSetup Object	671
		ParameterFieldDefinition Object	673

<p>ParameterFieldDefinitions Collection679 Preview method691 PrevPageButtonClicked event732 PrintButtonClicked event732 PrinterInfo Object681 PrintingStatus Object681 PrintOut method692, 721 PrintSetupButtonClicked event733 PrintWindowOptions Object683 PromptForExportOptions method641 ReadingRecords event694 ReadRecords method693 RefreshButtonClicked event733 RenderEPF method663 RenderHTML method663 RenderTotallerETF method665 RenderTotallerHTML method666 Report Object685 ReportObjects Collection696 ReportOptions Object697 ReportSummaryInfo Object699 Reset method641 SearchButtonClicked event734 SearchForText method669 Section Object700 SectionOptions Object701 Sections Collection705 SelectPrinter method693 SetCurrentValue method677 SetDefaultValue method678 SetLogOnInfo method632 SetPrivateData method633 SetSessionInfo method633 ShowFirstPage method721 ShowGroup event734 ShowLastPage method721 ShowNextPage method722 ShowNthPage method722 ShowPreviousPage method722 SortField Object706 SortFields Collection707 SpecialVarFieldDefinition Object708 Start event694</p>	<p>Stop event695 SubreportObject Object711 SummaryFieldDefinition Object712 SummaryFieldDefinitions Collection715 TestConnectivity method634 Text Object716 TrackCursorInfo Object717 Verify method624 View Object719 Views Collection723 Window Object724 ZoomLevelChanging event735 ZoomPreviewWindow method722 ReplaceSelectionFormula method602 ReplaceSelectionFormula method, VCL830 Report Engine API48 before using in your application45 distributing applications76 introduction to44 using46 Report Engine API REC Library111 structures66 using in Visual Basic78 variable length strings63 Report Engine Automation Server26 Report Object AddGroup method, REOL689 CancelPrinting method, REOL690 ClearError method, REOL690 DiscardSavedData method, REOL690 Export method, REOL691 OpenSubreport method, REOL691 Preview method, REOL691 PrintOut method, REOL692 ReadingRecords event, REOL694 ReadRecords method, REOL693 SelectPrinter method, REOL693 Start event, REOL694 Stop event, REOL695</p>	<p>Report Object, REOL685 report, web changing selection formulas in17 refreshing data17 ReportDisplayPage property556 ReportDisplayPage property, VCL798 ReportFileName property557 ReportLatestPage property558 ReportLatestPage property, VCL799 ReportName property, VCL799 ReportObjects Collection, REOL696 ReportOptions Object, REOL697 ReportPrinterSetup property, VCL800 reports monitoring web activity with20 reports, HTML limitations of39 reports, web linking to12 reports, web activity designing20 requesting from browser21 ReportSource property559 ReportStartPage property559 ReportStartPage property, VCL801 ReportSummaryInfo Object, REOL699 ReportTitle property560 ReportTitle property, VCL801 ReportWindowHandle method, VCL830 requesting web activity reports from your browser21 Reset method603 Reset method, REOL641 Resetting Array Properties tutorial, VCL837 Resetting Sort Array Properties tutorial, VCL838 Resetting TCRpe String Properties tutorial, VCL838 RetrieveDataFiles method603 RetrieveDataFiles method, VCL831 RetrieveLogonInfo method604 RetrieveLogonInfo method, VCL832 RetrieveSQLQuery method604</p>
--	--	---

<p>RetrieveSQLQuery method, VCL 832</p> <p>RetrieveStoredProcParams method 605</p> <p>RetrieveStoredProcParams method, VCL 832</p> <p>RetrieveTableData property, VCL 802</p> <p>Return Types 1068</p> <p>Returning User Defined Errors 1077</p> <p>runtime changing VCL properties 110</p> <p>runtime properties ActiveX Control 84</p>	<p>SetPrivateData method, REOL 633</p> <p>SetSessionInfo method, REOL 633</p> <p>Setting Up a UFL Project 1070</p> <p>ShowFirstPage method, REOL 721</p> <p>ShowFirstPage method, VCL 833</p> <p>ShowGroup event, REOL 734</p> <p>ShowLastPage method, REOL 721</p> <p>ShowLastPage method, VCL 834</p> <p>ShowNextPage method, REOL 722</p> <p>ShowNextPage method, VCL 834</p> <p>ShowNthPage method, REOL 722</p> <p>ShowPage method, VCL 835</p> <p>ShowPreviousPage method, REOL 722</p> <p>ShowPreviousPage method, VCL 835</p> <p>Smart Navigation 12</p> <p>Smart Viewer customizing 29</p> <p>Smart Viewer, Crystal 32</p> <p>Smart Viewer/ActiveX 34</p> <p>Smart Viewer/HTML 39</p> <p>Smart Viewer/Java 33</p> <p>Smart Viewers printing from 32</p> <p>SortField Object, REOL 706</p> <p>SortFields Collection Add method, REOL 707</p> <p>Delete method, REOL 708</p> <p>SortFields Collection, REOL 707</p> <p>SortFields property 568</p> <p>SortFields property, VCL 809</p> <p>SpecialVarFieldDefinition Object, REOL 708</p> <p>SpecifyDataSourceField method 605</p> <p>SQL databases 18</p> <p>SQL stored procedures 19</p> <p>SQLQuery property 569</p> <p>SQLQuery property, VCL 811</p> <p>Start event, REOL 694</p> <p>Status property 570</p> <p>Status property, VCL 812</p> <p>Stop event, REOL 695</p> <p>stored procedures, SQL 19</p> <p>Stored ProcParam property 571</p> <p>Stored ProcParam property, VCL 813</p> <p>structures COLORREF 353</p> <p>DEVMODE 353</p>	<p>PECharSepFileOptions 389</p> <p>PEDrillOnDetailEvent- Info 392</p> <p>PEDrillOnGroupEvent- Info 394</p> <p>PEEnableEventInfo 397</p> <p>PEExportOptions 399</p> <p>PEFieldValueInfo 407</p> <p>PEGeneralPrintWindow- EventInfo 409</p> <p>PEGraphDataInfo 410</p> <p>PEGraphOptions 413</p> <p>PEGraphTextInfo 415</p> <p>PEGroupOptions 418</p> <p>PEGroupTreeButtonClicked- EventInfo 421</p> <p>PEJobInfo 423</p> <p>PELogOnInfo 425</p> <p>PEParameterFieldInfo 427</p> <p>PEParameterInfo 432</p> <p>PEPrintFileOptions 434</p> <p>PEPrintOptions 435</p> <p>PEReadingRecordsEvent- Info 438</p> <p>PEReportSummaryInfo 440</p> <p>PESearchButtonClickedEvent- Info 442</p> <p>PESectionOptions 444</p> <p>PESessionInfo 448</p> <p>PEShowGroupEventInfo 450</p> <p>PEStartEventInfo 451</p> <p>PEStopEventInfo 453</p> <p>PESubreportInfo 455</p> <p>PETableLocation 457</p> <p>PETableType 459</p> <p>PETrackCursorInfo 461</p> <p>PEValueInfo 466</p> <p>PEWindowOptions 470</p> <p>PEZoomLevelChangingEvent- Info 473</p> <p>Report Engine API 66</p> <p>UXDDiskOptions 475</p> <p>UXDMAPIOptions 476</p> <p>UXDPostFolderOptions 479</p> <p>UXDSMIOptions 478</p> <p>UXDVIMOptions 481</p> <p>UXFCharSeparated- Options 482</p> <p>UXFCommaTabSeparated- Options 484</p> <p>UXFDIFOptions 485</p> <p>UXFHTML3Options 487</p> <p>UXFODBCOptions 488</p> <p>UXFPaginatedText- Options 489</p>
--	---	--

<p>UXFRecordStyle- Options 490</p> <p>SubreportObject Object, REOL 711</p> <p>SubreportToChange property 572</p> <p>SummaryFieldDefinition Object, REOL 712</p> <p>SummaryFieldDefinitions Collection, REOL 715</p> <p>T</p> <p>TCrpe Component, VCL 740</p> <p>TCrpePrinterSetup Class, VCL 740</p> <p>TermForJob Function 1076</p> <p>TestConnectivity method, REOL 634</p> <p>Text Object, REOL 716</p> <p>TrackCursorInfo Object, REOL 717</p> <p>U</p> <p>UFJOB Modules 1083</p> <p>UFJOB.C 1084</p> <p>UFJOB.H 1084</p> <p>UFL Function Implementation 1077</p> <p>User Function Libraries - An Overview 1065</p> <p>UserName property 574</p> <p>UserName property, VCL 813</p> <p>UserPrinterSetup.Enabled property, VCL 751</p> <p>using the ActiveX Control 84</p> <p>Using the Crystal Report Engine 46</p> <p>using the Crystal Report Engine API in Visual Basic 78</p> <p>using the VCL 109</p> <p>UXDDiskOptions C syntax 475</p> <p>Delphi syntax 476</p> <p>UXDDiskOptions structure 475</p> <p>UXDMAPIOptions C syntax 476</p> <p>Delphi syntax 477</p> <p>UXDMAPIOptions structure 476</p> <p>UXDPostFolderOptions C syntax 479</p> <p>Delphi syntax 480</p> <p>UXDPostFolderOptions structure 479</p> <p>UXDSMIOptions C syntax 478</p> <p>Delphi syntax 479</p>	<p>UXDSMIOptions structure 478</p> <p>UXDVIMOptions C syntax 481</p> <p>Delphi syntax 482</p> <p>UXDVIMOptions structure 481</p> <p>UXFCharSeparatedOptions C syntax 482</p> <p>Delphi syntax 483</p> <p>UXFCharSeparatedOptions structure 482</p> <p>UXFCommaTabSeparatedOptions C syntax 484</p> <p>Delphi syntax 485</p> <p>UXFCommaTabSeparatedOptions structure 484</p> <p>UXFDIFOptions C syntax 485, 486</p> <p>UXFDIFOptions structure 485</p> <p>UXFHML3Options C syntax 487</p> <p>Delphi syntax 487</p> <p>UXFHML3Options structure 487</p> <p>UXFODBCOptions C syntax 488</p> <p>Delphi syntax 488</p> <p>UXFODBCOptions structure 488</p> <p>UXFPaginatedTextOptions C syntax 489</p> <p>Delphi syntax 490</p> <p>UXFPaginatedTextOptions structure 489</p> <p>UXFRecordStyleOptions C syntax 490</p> <p>Delphi syntax 491</p> <p>UXFRecordStyleOptions structure 490</p> <p>V</p> <p>variable length strings 63</p> <p>VB syntax DEVMODE 370</p> <p>PECancelPrintJob 121</p> <p>PECanCloseEngine 122</p> <p>PECheckFormula 124</p> <p>PECheckGroupSelection- Formula 125</p> <p>PECheckSelection- Formula 127</p> <p>PECloseButtonClicked- EventInfo 391</p> <p>PECloseEngine 128</p> <p>PEClosePrintJob 129</p> <p>PECloseSubreport 131</p> <p>PECloseWindow 132</p> <p>PEConvertPFIInfoToVInfo 134</p> <p>PEConvertVInfoToPFIInfo 135</p> <p>PEDeleteNthGroupSort- Field 137</p> <p>PEDeleteNthSortField 139</p> <p>PEDiscardSavedData 140</p> <p>PEDrillOnGroupEvent- Info 395</p> <p>PEEnableEvent 142</p> <p>PEEnableEventInfo 398</p> <p>PEEnableProgressDialog 143</p> <p>PEExportOptions 405</p> <p>PEExportPrintWindow 145</p> <p>PEExportTo 146</p> <p>PEFieldValueInfo 408</p> <p>PEGeneralPrintWindow- EventInfo 409</p> <p>PEGetAreaFormat 148</p> <p>PEGetAreaFormat- Formula 150</p> <p>PEGetEnableEventInfo 151</p> <p>PEGetErrorCode 153</p> <p>PEGetErrorText 154</p> <p>PEGetExportOptions 155</p> <p>PEGetFormula 157</p> <p>PEGetGraphData 159</p> <p>PEGetGraphOptions 160</p> <p>PEGetGraphText 162</p> <p>PEGetGraphType 163</p> <p>PEGetGroupCondition 166</p> <p>PEGetGroupOptions 167</p> <p>PEGetGroupSelection- Formula 169</p> <p>PEGetHandleString 171</p> <p>PEGetJobStatus 172</p> <p>PEGetMargins 174</p> <p>PEGetMinimumSection- Height 175</p> <p>PEGetNDetailCopies 177</p> <p>PEGetNFormulas 178</p> <p>PEGetNGroups 179</p> <p>PEGetNGroupSortFields 181</p> <p>PEGetNPages 182</p> <p>PEGetNParameterFields 183</p> <p>PEGetNParams 185</p> <p>PEGetNSections 186</p> <p>PEGetNSortFields 187</p> <p>PEGetNSubreportsIn- Section 189</p> <p>PEGetNTables 190</p> <p>PEGetNthFormula 192</p> <p>PEGetNthGroupSortField 194</p> <p>PEGetNthParam 196</p> <p>PEGetNthParameterField 198</p> <p>PEGetNthParamInfo 199</p> <p>PEGetNthSortField 201</p>
--	---

PEGetNthSubreportInfo
 Section 203
 PEGetNthTableLocation 204
 PEGetNthTableLogOn-
 Info 205
 PEGetNthTableSession-
 Info 207
 PEGetNthTableType 208
 PEGetPrintDate 210
 PEGetPrintOptions 212
 PEGetReportTitle 213, 215
 PEGetSectionCode 216
 PEGetSectionFormat 218
 PEGetSectionFormat-
 Formula 221
 PEGetSelectedPrinter 223
 PEGetSelectionFormula 225
 PEGetSQLQuery 227
 PEGetSubreportInfo 228
 PEGetTrackCursorInfo 230
 PEGetVersion 231
 PEGetWindowHandle 233
 PEGetWindowOptions 235
 PEGraph DataInfo 412
 PEGraph Options 414
 PEGraph TextInfo 416
 PEGroupOptions 420
 PEGroupTreeButtonClicked-
 EventInfo 422
 PEHasSavedData 238
 PEIsPrintJobFinished 239
 PEJobInfo 424
 PELogOffServer 241
 PELogOnInfo 426
 PELogOnServer 242
 PELogOnSQLServerWith-
 PrivateInfo 245
 PENextPrintWindow-
 Magnification 246
 PEOpenEngine 247
 PEOpenPrintJob 249
 PEOpenSubreport 251
 PEOutputToPrinter 256
 PEOutputToWindow 260
 PEParameterInfo 430, 433
 PEPrintControlsShowing 262
 PEPrintOptions 437
 PEPrintReport 264
 PEPrintWindow 266
 PEReadingRecordsEvent-
 Info 439
 PEReportSummaryInfo 441
 PESearchButtonClickedEvent-
 Info 443
 PESectionOptions 446
 PESelectPrinter 268

PESessionInfo 449
 PESetAreaFormat 270
 PESetAreaFormat-
 Formula 273
 PESetDialogParent-
 Window 274
 PESetEventCallback 281
 PESetFont 286
 PESetFormula 288
 PESetGraphData 290
 PESetGraphOptions 292
 PESetGraphText 293
 PESetGraphType 296
 PESetGroupCondition 300
 PESetGroupOptions 303
 PESetGroupSelection-
 Formula 304
 PESetMargins 306
 PESetMinimumSection-
 Height 308
 PESetNDetailCopies 310
 PESetNthGroupSortField 312
 PESetNthParam 314
 PESetNthParameterField 316
 PESetNthSortField 318
 PESetNthTableLocation 319
 PESetNthTableSession-
 Info 323
 PESetPrintDate 325
 PESetPrintOptions 327
 PESetReportSummary-
 Info 328
 PESetReportTitle 330
 PESetSectionFormat 332
 PESetSectionFormat-
 Formula 335
 PESetSelectionFormula 337
 PESetSQLQuery 338
 PESetTrackCursorInfo 340
 PESetWindowOptions 341
 PEShowNthPage 345
 PEShowPrintControls 347
 PEStartEventInfo 452
 PEStartPrintJob 349
 PEStopEventInfo 454
 PESubreportInfo 456
 PETableLocation 458
 PETableType 460
 PETestNthTable-
 Connectivity 350
 PETrackCursorInfo 464
 PEValueInfo 468
 PEWindowOptions 471
 PEZoomLevelChanging-
 EventInfo 474
 PEZoomPreviewWindow 352

VCL
 Access Notes Sparse Array
 Properties tutorial 837
 Action property 740
 adding to project 108
 ChangeTableLocation
 method 824
 changing properties 110
 Clear method 825
 Close Window method 825
 Connect property 741
 CopiesToPrinter
 property 742
 DataFiles property 743
 DataFilesLocation
 property 744
 Destination property 744
 DetailCopies property 745
 DialogParentHandle
 property 746
 DiscardSavedData
 property 747
 EMailCCList property 747
 EMailMessage property 748
 EMailSubject property 749
 EMailToList property 749
 EMailVIMBCCLIST
 property 750
 ExchangeFolder property 751
 ExchangePassword
 property 752
 ExchangeProfile property 753
 Execute property 753
 ExportPrintWindow
 method 826
 FetchSelectionFormula
 method 826
 Focused method 827
 Formulas property 754
 GraphData property 755
 GraphOptions property 757
 GraphText property 759
 GraphType property 760
 GroupCondition
 property 761
 GroupSelectionFormula
 property 763
 GroupSortFields property 764
 LastErrorNumber
 property 765
 LastErrorString property 766
 LogoffServer method 827
 LogOnInfo property 767
 LogonServer method 828
 MarginBottom property 768
 MarginLeft property 769

MarginRight property.....	770	ReportPrinterSetup property	800	WindowMaxButton property	818
MarginTop property.....	770	ReportStartPage property....	801	WindowMinButton property	818
MDIChild property	771	ReportTitle property	801	WindowParentHandle property	819
NumberOfGroups property	772	ReportWindowHandle method	830	WindowState property.....	820
OnExecute event	836	Resetting Array Properties tutorial	837	WindowTitle property	821
OnLoadDataFiles event	836	Resetting Sort Array Properties tutorial	838	WindowTop property	821
PageCount method	829	Resetting TCrpe String Properties tutorial	838	WindowWidth property....	822
ParamFields property.....	772	RetrieveDataFiles method.....	831	ZoomMagnification property	823
Password property	773	RetrieveLogOnInfo method.....	832	ZoomPreviewWindow property	823
PrintControls property	774	RetrieveSQLQuery method.....	832	Verify method, REOL	624
PrintDay property	774	RetrieveStoredProcParams method.....	832	View Object Close method, REOL	720
PrintEnded property.....	775	RetrieveTableData property	802	Export method, REOL	721
PrinterCollation property	776	SectionFont property.....	803	NextMagnification method, REOL	721
PrinterCopies property.....	777	SectionFormat property	804	PrintOut method, REOL....	721
PrinterDriver property.....	777	SectionLineHeight property	806	ShowFirstPage method, REOL	721
PrinterMode property	779	SectionMinHeight property	807	ShowLastPage method, REOL	721
PrinterName property	780	SelectionFormula property	808	ShowNextPage method, REOL	722
PrinterPort property	781	SetFocus method.....	833	ShowNthPage method, REOL	722
PrinterStartPage property	783	ShowFirstPage method.....	833	ShowPreviousPage method, REOL	722
PrinterStopPage property	783	ShowLastPage method	834	ZoomPreviewWindow method, REOL	722
PrintFileCharSepQuote property	784	ShowNextPage method	834	View Object, REOL.....	719
PrintFileCharSepSeparator property	785	ShowPage method	835	Viewing the Crystal Report Engine Object Library.....	89
PrintFileLinesPerPage property	785	ShowPreviousPage method.....	835	Views Collection, REOL.....	723
PrintFileName property	786	SortFields property	809	Visual Basic using the Crystal Report Engine API in.....	78
PrintFileODBCPassword property	788	SQLQuery property.....	811	Visual Component Library adding to project.....	108
PrintFileODBCSource property	789	Status property	812	changing properties	110
PrintFileODBCTable property	789	StoredProcParam property	813	using	109
PrintFileODBCUser property	790	TCrpe Component	740	Visual InterDev Design-time ActiveX Control.....	26
PrintFileType property	791	TCrpePrinterSetup Class	740		
PrintFileUseRptDateFmt property	793	UserName property.....	813		
PrintFileUseRptNumberFmt property	793	UserPrinterSetup.Enabled property	751		
PrintMonth property	794	using.....	109		
PrintWindow method	829	WindowBorderStyle property	814		
PrintYear property	795	WindowControlBox property	815		
ProgressDialog property	796	WindowControls property	816		
RecordsPrinted property	796	WindowHeight property ...	816		
RecordsRead property	797	WindowLeft property	817		
RecordsSelected property	798				
ReplaceSelectionFormula method	830				
ReportDisplayPage property	798				
ReportLatestPage property	799				
ReportName property	799				

web report data refreshing.....	17	
Web Report Server parameters.....	13	
web reports changing selection formulas in	17	
linking to	12	
web site sample	30	
Window Object		
ActivatePrintWindow event, REOL	725	
CancelButtonClicked event, REOL	725	
Close method, REOL.....	724	
CloseButtonClicked event, REOL	726	
ClosePrintWindow event, REOL	726	
DeactivatePrintWindow event, REOL	727	
DrillOnDetail event, REOL	727	
DrillOnGroup event, REOL	728	
ExportButtonClicked event, REOL	729	
FirstPageButtonClicked event, REOL	729	
GroupTreeButtonClicked event, REOL.....	730	
LastPageButtonClicked event, REOL	730	
NextPageButtonClicked event, REOL	731	
PrevPageButtonClicked event, REOL	732	
PrintButtonClicked event, REOL	732	
PrintSetupButtonClicked event, REOL.....	733	
RefreshButtonClicked event, REOL	733	
SearchButtonClicked event, REOL	734	
ShowGroup event, REOL...734		
ZoomLevelChanging event, REOL	735	
Window Object, REOL.....724		
WindowAllowDrillDown property	574	
WindowBorderStyle property	575	
WindowBorderStyle property, VCL.....814		
WindowControlBox property ...	576	
WindowControlBox property, VCL	815	
WindowControls property	577	
WindowControls property, VCL	816	
WindowHeight property	578	
WindowHeight property, VCL	816	
WindowLeft property	578	
WindowLeft property, VCL	817	
WindowMaxButton property ...	579	
WindowMaxButton property, VCL	818	
WindowMinButton property	580	
WindowMinButton property, VCL	818	
WindowParentHandle property	581	
WindowParentHandle property, VCL	819	
WindowShowCancelBtn property	582	
WindowShowCloseBtn property	583	
WindowShowExportBtn property	583	
WindowShowGroupTree property	584	
WindowShowNavigationCtls property	585	
WindowShowPrintBtn property	586	
WindowShowPrintSetupBtn property	586	
WindowShowProgressCtls property	587	
WindowShowRefreshBtn property	588	
WindowShowSearchBtn property	589	
WindowShowZoomCtl property	590	
WindowState property	590	
WindowState property, VCL820		
WindowTitle property.....591		
WindowTitle property, VCL821		
WindowTop property	592	
WindowTop property, VCL.....821		
WindowWidth property.....593		
WindowWidth property, VCL	822	
Working with section codes....116		

Z

ZoomLevelChanging event, REOL.....735	
ZoomMagnification property, VCL.....823	
ZoomPreviewWindow property, VCL.....823	