Seagate Crystal Reports™ 8
Technical Reference Supplemental Guide

Seagate Software, Inc.
915 Disc Drive
Scotts Valley
California, USA 95066

Trademark Acknowledgements

CONTENTS

1 The Crystal Report Engine Class Library Reference

What you will find in this chapter...

THE CRYSTAL REPORT ENGINE CLASS LIBRARY

Introduction

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 the following 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 PEPLUS.H 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. PEPLUS.CPP 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.

- CRPE32M.LIB

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.

Classes in the Crystal Report Engine Class Library

The Crystal REC Library consists of two primary classes:

1. The *class CRPEngine, Page 3*, 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.

2. The *class CRPEJob, Page 12*, 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.

CLASS CRPENGINE

This section describes the class CRPEngine:publicCObject and methods.

class CRPEngine : public CObject

The CRPEngine class is designed so that there should be only one CRPEngine object in the entire application. The CRPEngine object contains methods that are common to all print jobs (SQL connections, version information, etc.). More importantly, CRPEngine 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 that you want to open. If successful, you will be returned a pointer to a CRPEJob object.

Constructor CRPEngine::CRPEngine

CRPEngine::CRPEngine is the constructor for the class. If the open parameter 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 10*.

Constructor Syntax

```
CRPEngine ( BOOL open );
```

Parameter

open	Indicates whether the Crystal Report Engine should be opened when the CRPEngine object is created. The default for this parameter is FALSE.

class CRPEngine Methods

The following methods are discussed in this section.

CRPEngine::AddJob

Use CRPEngine::AddJob to add a new job object.

Syntax

```
void AddJob ( CRPEJob *job );
```

Parameter

job	Specifies a pointer to the CRPEJob object.

CRPEngine::CanClose

Use CRPEngine::CanClose to determine whether the Seagate Crystal Report Engine is busy. Errors can occur in your application or on a system if you attempt to close the Seagate Crystal Report Engine while it is processing a print job. To determine whether the Report Engine can be closed safely in an application, use this method before attempting to close the Seagate Crystal Report Engine (for example, when the user tries to exit).

Syntax

```
BOOL CanClose ();
```

Returns

- TRUE if the Seagate Crystal Report Engine can be closed.
- FALSE if Seagate Crystal Report Engine is still busy.

CRPEngine::Close

Use CRPEngine::Close to close the Seagate Crystal Report Engine.

Syntax

```
void Close ();
```

CRPEngine::GetEngine

Use CRPEngine::GetEngine to retrieve a pointer to the current CRPEngine object being used in the application.

Syntax

```
static CRPEngine *GetEngine ();
```

Returns

Returns a pointer to the current CRPEngine object.

CRPEngine::GetEngineStatus

Use CRPEngine::GetEngineStatus to get the current engine status.

Syntax

```
static Status GetEngineStatus ();
```

Returns

Returns one of the following values of Status enumerated type indicating the current status of the Seagate Crystal Report Engine.

engineOpen	The Crystal Report Engine is currently open.
engineClosed	The Crystal Report Engine is not yet open, or has been closed.
engineMissing	No Crystal Report Engine is available. Make sure the Crystal Report Engine DLL is located in a directory that appears in your PATH.

CRPEngine::GetErrorCode

Use CRPEngine::GetErrorCode to retrieve the current error code of the Seagate Crystal Report Engine. When a call to another function fails, this call retrieves the error code that was generated so that you can take some action based on that error code.

Syntax

```
short GetErrorCode ();
```

Returns

- Returns the current Seagate Crystal Report Engine **Error Codes** (page 159).
- Returns 0 (zero) if no error has occurred.

CRPEngine::GetErrorText

Use CRPEngine::GetErrorText to retrieve a descriptive Seagate Crystal Report Engine error message.

Syntax

```
CString GetErrorText ();
```

Returns

Returns a CString object containing the text for the current error.

CRPEngine::GetHandleString

Use CRPEngine::GetHandleString to retrieve the text to which the string handle is pointing. The CString object will contain 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 GetHandleString (
          HANDLE textHandle,
          short textLength,
          CString &string );
```

Parameters

textHandle	Specifies the handle of the string containing the text of interest.
textLength	Specifies the length of the text string, including the terminating null byte. This value should be identical to the length of the string obtained by the variable length string function.
string	Reference to the CString object obtained by the variable length string function.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Because the string handle is discarded once the method is called, you can use this call only once with a given string handle. If you expect to use the string later, you will need to save it.
- When you call the method 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.

CRPEngine::GetNPrintJobs

Use CRPEngine::GetNPrintJobs to retrieve the number of print job (CRPEJob) objects that currently exist for the current CRPEngine object.

Syntax

```
int GetNPrintJobs ();
```

Returns

- Returns the number of CRPEJob objects currently open.
- Returns -1 if an error occurs.

CRPEngine::GetVersion

Use CRPEngine::GetVersion to retrieve the version number of the Seagate Crystal Report Engine DLL (CRPE32.DLL) or the version number of the Seagate 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 Seagate Crystal Report Engine and you need to verify the version number first. This function can be a safeguard for users who have more than one version of the Seagate Crystal Report Engine with the older version appearing earlier in the path than the newer version.

Syntax

```
short GetVersion ( short versionRequested );
```

Parameters

versionRequested	Specifies whether the version number of the Seagate Crystal Report Engine or the Seagate Crystal Report Engine DLL is being requested. Use one of the following constants.	
	Constant	***Description***
	PEP_GV_DLL	Returns the version of the DLL.
	PEP_GV_ENGINE	Returns the version of the Crystal Report Engine.

Returns

Returns the version number of the currently used Seagate Crystal Report Engine or Seagate Crystal Report Engine DLL.

CRPEngine::LogOffServer

Use CRPEngine::LogOffServer to close an existing connection to an SQL server. Connection information is provided through **CRPELogOnInfo** (page 128).

Syntax

```
BOOL LogOffServer (
          const _TCHAR *dllName,
          const CRPELogOnInfo *logOnInfo );
```

Parameters

dllName	Specifies a pointer to the name of the Seagate Crystal Reports DLL for the server or password protected non-SQL table from which you want to log off.
logOnInfo	Specifies a pointer to **CRPELogOnInfo** (page 128).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEngine::LogOnServer

Use CRPEngine::LogOnServer to open a connection to an SQL server. More than one print job may use this connection to the SQL server. Information required to establish a connection is provided through **CRPELogOnInfo** (page 128).

Syntax

```
BOOL LogOnServer (
          const _TCHAR *dllName,
          const CRPELogOnInfo *logOnInfo );
```

Parameters

dllName	Specifies a pointer to the name of the Seagate Crystal Reports DLL for the server or password-protected non-SQL table to which you want to log on.
logOnInfo	Specifies a pointer to **CRPELogOnInfo** (page 128).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEngine::LogOnSQLServerWithPrivateInfo

Use CRPEngine::LogOnSQLServerWithPrivateInfo to allow the Seagate 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 LogOff call from disconnecting an application's existing connection to the Server.

Syntax

```
BOOL LogOnSQLServerWithPrivateInfo (
          const _TCHAR *dllName,
          void *privateInfo );
```

Parameters

dllName	Specifies the name of the 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 in Developer's Online Help.
privateInfo	Specifies a pointer to 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- This method can be used only with database connections established by ODBC or Q+E Library 2.0. Any other database connections cannot 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).

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).

Call	*Description*
qeConnect	Opens a connection to the server.
qeGetODBCHdbc	Returns the ODBC HDBC.

CRPEngine::Open

Use CRPEngine::Open to open the Seagate Crystal Report Engine. This method is necessary only 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.

Syntax

```
BOOL Open ();
```

Returns

- TRUE if the Crystal Report Engine was opened successfully.
- FALSE if the call failed.

CRPEngine::OpenJob

Use CRPEngine::OpenJob to open the report specified by reportFileName. A pointer to a CRPEJob object is returned. Print job attributes may be referenced through this object.

Syntax

```
CRPEJob *OpenJob ( const _TCHAR *reportFileName );
```

Parameter

reportFileName	Specifies a pointer to the name and path (if necessary) of the report file being opened for the specified print job.

Returns

Returns a pointer to a CRPEJob object for the opened print job.

CRPEngine::PrintReport

Use CRPEngine::PrintReport to provide a quick but limited way to print a report. The report may be sent only to a printer or preview window or both. Use this class method any time that you simply want to print a report from an application without giving the user the ability to customize the report.

Syntax

```
short PrintReport (
          const _TCHAR *reportFilePath,
          BOOL toPrinter,
          BOOL toWindow,
          const _TCHAR *title,
          int left,
          int top,
          int width,
          int height,
          DWORD style,
          CWnd *parentWindow );
```

Parameters

reportFilePath	Specifies a pointer to the filename and path of the report to be printed.
toPrinter	Specifies whether the report is to be sent to the default printer.
toWindow	Specifies whether the report is to be displayed in a preview window.
title	Specifies a pointer to the title string that 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. The 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.

Returns

- Returns 0 (zero) if the call is successful.
- Returns an error code if an error occurs.

CRPEngine::RemoveJob

Use CRPEngine::RemoveJob to remove a job object.

Syntax

```
void RemoveJob ( CRPEJob *job );
```

Parameter

job	Specifies a pointer to the CRPEJob object that you want to remove.

CLASS CRPEJOB

This section describes the class CRPEJob public CObject and methods.

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 member function *CRPEngine::OpenJob, Page 10*, specifying the report file name to open. If successful, you will be returned a pointer to the 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 these methods correspond to similar functions available in the Crystal Report Engine API.

Constructor CRPEJob::CRPEJob

- This class constructor is called automatically by *CRPEngine::OpenJob, Page 10.* Do not call this constructor from within your own code.
- This class constructor also stores the job number internally for future use in Seagate Crystal Report Engine API calls.

Constructor Syntax

```
CRPEJob ( short jobHandle );
CRPEJob (  short jobHandle,
           CRPEJob *parentJob );
```

class CRPEJob Methods

The following methods are discussed in this section, arranged alphabetically.

class CRPEJob Methods A...F

class CRPEJob Methods G (Get...)

class CRPEJob Methods H...Z

CRPEJob::AddParameterCurrentRange

Use CRPEJob::AddParameterCurrentRange to add a parameter range to the specified parameter field of a report.

Syntax

```
BOOL AddParameterCurrentRange (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEValueInfo *rangeStart,
          CRPEValueInfo *rangeEnd,
          short rangeInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.

rangeStart	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the new lower bound of the value range.
rangeEnd	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the new upper bound of the value range.
rangeInfo	Use this bit mapped value to indicate whether the upper and/or lower bound of the range should be included. Set rangeInfo to one of the following constants.

Constant	*Value*	*Description*
PEP_RI_NOUPPERBOUND	4	*rangeStart* must contain valid information. Set *rangeEnd* to NULL.
PEP_RI_NOLOWERBOUND	8	*rangeEnd* must contain valid information. Set *rangeStart* to NULL.
PEP_RI_INCLUDEUPPERBOUND OR PEP_RI_INCLUDELOWERBOUND	3	Both *rangeStart* and *rangeEnd* must contain valid information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Set *rangeStart* to NULL to retrieve maximum value only. *rangeStart* must be non-NULL if *rangeEnd* is NULL.
- Set *rangeEnd* to NULL to retrieve minimum value only. *rangeEnd* must be non-NULL if *rangeStart* is NULL.
- If both *rangeStart* and *rangeEnd* are specified (that is, non-NULL), then the valueType member of **CRPEValueInfo** (page 152) for rangeStart and rangeEnd must be the same or an error code PEP_ERR_INCONSISTANTTYPES will be returned.

CRPEJob::AddParameterCurrentValue

Use CRPEJob::AddParameterCurrentValue to add a current value to the specified parameter field in the specified report.

Syntax

```
BOOL AddParameterCurrentValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEValueInfo *currentValue );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
currentValue	Specifies a pointer to the **CRPEValueInfo** (page 152), which will contain the current value to be added to the parameter field.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::AddParameterDefaultValue

Use CRPEJob::AddParameterDefaultValue to add a default value to a specified parameter field in a specified report.

Syntax

```
BOOL AddParameterDefaultValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEValueInfo *valueInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
valueInfo	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the default value to be added to the parameter field.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::Cancel

Use CRPEJob::Cancel to cancel processing of a print job. This method can be tied to a control that allows the user to cancel a print job in progress.

Syntax

```
void Cancel ();
```

CRPEJob::CheckFormula

Use CRPEJob::CheckFormula to check a formula for validity. This method works just like the Check button in Seagate Crystal Reports Formula Editor. If the named formula contains an error, the method returns FALSE.

Syntax

```
BOOL CheckFormula ( const _TCHAR *formulaName );
```

Parameter

formulaName	Specifies a pointer to the name of the formula that you want to check for errors.

Returns

- TRUE if the formula text is correct.
- FALSE if the checked formula contains an error or if the call fails.

CRPEJob::CheckGroupSelectionFormula

Use CRPEJob::CheckGroupSelectionFormula to check the group selection formula for the report for errors. This method works just like the Check button in Seagate Crystal Reports Formula Editor. If the group selection formula contains an error, the method returns FALSE.

Syntax

```
BOOL CheckGroupSelectionFormula ();
```

Returns

- TRUE if the formula text is correct.
- FALSE if the checked formula contains an error or if the call fails.

CRPEJob::CheckNthTableDifferences

Use CRPEJob::CheckNthTableDifferences to retrieve information about table differences.

Syntax

```
BOOL CheckNthTableDifferences (
          short tableN,
          CRPETableDifferenceInfo *tableDifferenceInfo );
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to retrieve change information.
tableDifferenceInfo	Specifies a pointer to **CRPETableDifferenceInfo** (page 147), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::CheckSelectionFormula

Use CRPEJob::CheckSelectionFormula to check the record selection formula for the report for errors. This method works just like the Check button in Seagate Crystal Reports Formula Editor. If the selection formula contains an error, the method returns FALSE.

Syntax

```
BOOL Check SelectionFormula ();
```

Returns

- TRUE if the formula text is correct.
- FALSE if the checked formula contains an error or if the call fails.

CRPEJob::CheckSQLExpression

Use CRPEJob::CheckSQLExpression to check the specified SQL expression for errors. If the SQL expression contains an error, the method returns FALSE.

Syntax

```
BOOL CheckSQLExpression ( const _TCHAR *expressionName );
```

Parameter

expressionName	Specifies a pointer to the name of the SQL expression that you want to check.

Returns

- TRUE if the formula text is correct.
- FALSE if the checked formula contains an error or if the call fails.

CRPEJob::ClearParameterCurrentValuesAndRanges

Use CRPEJob::ClearParameterCurrentValuesAndRanges to clear the specified parameter field of all current values and ranges.

Syntax

```
BOOL ClearParameterCurrentValuesAndRanges (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::CloseSubreport

Use CRPEJob::CloseSubreport to close a subreport.

Syntax

```
BOOL CloseSubreport ();
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::Close

Use CRPEJob::Close to close the print job and delete the CRPEJob object. This method calls the destructor for the CRPEJob object. Printing will continue if it has not completed when this method is called. If the preview window is open, it stays open.

Syntax

```
void Close ();
```

CRPEJob::CloseWindow

Use CRPEJob::CloseWindow to close the preview window. If you are customizing preview window controls, implement this method to allow the user to close the preview window when finished with viewing the report.

Syntax

```
void CloseWindow ();
```

CRPEJob::ConvertPFInfoToVIInfo

Use to CRPEJob::ConvertPFInfoToVIInfo convert parameter field info to value info.

Syntax

```
BOOL ConvertPFInfoToVInfo (
     void FAR * value,
     short valueType,
     CRPEValueInfo *valueInfo );
```

Parameters

value	Specifies a pointer to the value to be converted.
valueType	Specifies the value type.
valueInfo	Specifies a pointer to **CRPEValueInfo** (page 152).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ConvertVIInfoToPFInfo

Use to CRPEJob::ConvertVIInfoToPFInfo convert value info to parameter field info.

Syntax

```
BOOL ConvertVInfoToPFInfo (
     CRPEValueInfo *valueInfo,
     WORD *valueType,
     void *value );
```

Parameters

valueInfo	Specifies a pointer to **CRPEValueInfo** (page 152).
valueType	Specifies a pointer to the value type.
value	Specifies a pointer to the value to be converted.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::DeleteNthGroupSortField

Use CRPEJob::DeleteNthGroupSortField to delete 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.

Syntax

```
BOOL DeleteNthGroupSortField ( short sortFieldN );
```

Parameter

sortFieldN	Specifies the number of the group sort field that you want to delete. The first group sort field added to the report is field 0, the second is 1, etc.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::DeleteNthParameterDefaultValue

Use CRPEJob::DeleteNthParameterDefaultValue to remove the indicated default value from the specified parameter field in the specified report.

Syntax

```
BOOL DeleteNthParameterDefaultValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
index	Specifies the index of the default value to remove.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::DeleteNthSortField

Use CRPEJob::DeleteNthSortField to remove 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.

Syntax

```
BOOL DeleteNthSortField ( short sortFieldN );
```

Parameter

sortFieldN	Specifies the number of the sort field that you want to delete. The first sort field added to the report is field 0, the second is 1, etc.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::DiscardSavedData

Use CRPEJob::DiscardSavedData to discard data that was previously saved with the report. If a report has been saved with data, you can use this method to discard the saved data, forcing the Seagate Crystal Report Engine to retrieve new data when the report is printed.

Syntax

```
BOOL DiscardSavedData ();
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::EnableEvent

Use CRPEJob::EnableEvent to enable/disable the group event specified by **CRPEEnableEventInfo** (page 109). All events are disabled by default.

Syntax

```
BOOL EnableEvent ( const CRPEEnableEventInfo *enableEventInfo );
```

Parameter

enableEventInfo	Specifies a pointer to **CRPEEnableEventInfo** (page 109).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::EnableProgressDialog

Use CRPEJob::EnableProgressDialog to enable/disable 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, etc.).

Syntax

```
BOOL EnableProgressDialog ( BOOL enable );
```

Parameter

enable	Specifies whether the progress dialog box is enabled. If this parameter is set to TRUE, the progress dialog box is enabled; if set to FALSE, the dialog box is disabled.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ExportPrintWindow

Use CRPEJob::ExportPrintWindow to export 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, such as a button click, menu command, etc.).

Syntax

```
BOOL ExportPrintWindow ( BOOL toMail );
```

Parameter

toMail	Specifies whether 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ExportTo

Use CRPEJob::ExportTo to set 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 103*.

Syntax

```
BOOL ExportTo ( const CRPEExportOptions *options );
```

Parameter

options	Specifies a pointer to **CRPEExportOptions** (page 110).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::FreeDevMode

Use CRPEJob::FreeDevMode to return the memory associated with the specified DEVMODE Microsoft Windows structure to the heap. The DEVMODE structure must have been retrieved from CRPEJob::SelectPrinter, Page 73, or CRPEJob::GetSelectedPrinter, Page 64.

Syntax

```
BOOL FreeDevMode ( DEVMODEA *mode );
```

Parameter

mode	Specifies a pointer to Microsoft Windows structure **DEVMODE** (*Seagate Crystal Reports Technical Reference Guide*).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetAllowPromptDialog

Use CRPEJob::GetAllowPromptDialog to determine if prompting for parameter values is allowed during printing for this job.

Syntax

```
BOOL GetAllowPromptDialog ();
```

Returns

Returns a Boolean value indicating whether prompting for parameter values is allowed during printing of a job. TRUE indicates that prompting is allowed.

CRPEJob::GetAreaFormat

Use CRPEJob::GetAreaFormat to set format information for the specified area.

Syntax

```
BOOL GetAreaFormat (
          short areaCode,
          CRPESectionOptions *options );
```

Parameters

areaCode	Specifies the area code for the area for which you want to retrieve format information.
options	Specifies a pointer to **CRPESectionOptions** (page 142), which will contain the area format information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetAreaFormatFormula

Use CRPEJob::GetAreaFormatFormula to retrieve the text of a conditional area format formula. Use this method in order to update the conditional area format formula and pass the changes back using *CRPEJob::SetAreaFormatFormula, Page 74*.

Syntax

```
BOOL GetAreaFormatFormula(
          short areaCode,
          short formulaName,
          CString &formulaText );
```

Parameters

areaCode	Specifies the code for the report area for which you want to set formatting options. For information on area codes, see *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.
formulaName	Specifies the name of the formatting formula for which you want to supply a new string. Use one of the PEP_FFN_XXX **Area/Section Format Formula Constants** (page 157).
formulaText	Reference to the CString containing the new formula text.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetEnableEventInfo

Use CRPEJob::GetEnableEventInfo to retrieve information related to enabled/disabled events from **CRPEEnableEventInfo** (page 109).

Syntax

```
BOOL EnableEvent ( const CRPEEnableEventInfo *enableEventInfo );
```

Parameter

enableEventInfo	Specifies a pointer to **CRPEEnableEventInfo** (page 109), which contains the information that you want to retrieve.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetErrorCode

Use CRPEJob::GetErrorCode to retrieve the error code for the print job. When a call to another method fails, this call retrieves the error code that was generated so that you can take some action based on that error code.

Syntax

```
short GetErrorCode ();
```

Returns

- Returns a Seagate Crystal Reports Class Library **Error Codes** (page 159).
- Returns 0 (value of constant PEP_NOERROR) if no error has occurred.

CRPEJob::GetErrorText

Use CRPEJob::GetErrorText to retrieve the descriptive error message for the print job.

Syntax

```
CString GetErrorText ();
```

Returns

Returns a CString object containing the text description of the current Seagate Crystal Reports Class Library **Error Codes** (page 159), if an error has occurred.

CRPEJob::GetExportOptions

Use CRPEJob::GetExportOptions to get export options from the user before exporting the report. CRPEGetExportOptions can be used to present a series of dialog boxes that retrieve export options from your users. These options are used by the Seagate Crystal Report Engine to fill in **CRPEExportOptions** (page 110). Then the CRPEJob::ExportTo, Page 28, can be used to set the print job destination using the information in the CRPEExportOptions structure.

Syntax

```
BOOL GetExportOptions ( CRPEExportOptions *options );
```

Parameter

options	Specifies a pointer to **CRPEExportOptions** (page 110).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetFieldMappingType

Use CRPEJob::GetFieldMappingType to retrieve the field mapping type code.

Syntax

```
BOOL GetFieldMappingType ( WORD mappingType );
```

Parameter

mappingType	Specifies a pointer to the field mapping type code, which should be one of the following constants.	
	Constant	***Description***
	PEP_FM_AUTO_FLD_MAP	Automatic field name mapping.
	PEP_FM_CRPEP_PROMPT_FLD_MAP	CRPE provides dialog box to map field manually.
	PEP_FM_EVENT_DEFINED_FLD_MAP	CRPE provides list of fields in the report and the new database.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetFormula

Use CRPEJob::GetFormula to return the formula text for the specified formula. CRPEJob::GetFormula is often used with *CRPEJob::SetFormula, Page 79,* to identify and then change an existing formula in response to a user selection at print time.

Syntax

```
BOOL GetFormula (
          const _TCHAR *formulaName,
          CString &formulaText );
```

Parameters

formulaName	Specifies a pointer to the name of the formula for which you want to retrieve the formula string.
formulaText	Reference to the CString object loaded with the specified formula text when CRPEJob::GetFormula is completed successfully.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetFormulaSyntax

Use CRPEJob::GetFormulaSyntax to retrieve the syntax used by the formula addressed in the last formula API call.

Syntax

```
BOOL GetFormulaSyntax ( CRPEFormulaSyntax *formulaSyntax );
```

Parameters

Parameter	*Description*
formulaSyntax	Specifies a pointer to **CRPEFormulaSyntax** (page 117), which will contain the information that you want to retrieve.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

For running total condition formula:

- formulaSyntax[0] is the syntax for the evalFormula, and
- formulaSyntax[1] is the syntax for the resetFormula.

CRPEJob::GetGraphAxisInfo

Use CRPEJob::GetGraphAxisInfo to retrieve chart axis information for the specified chart.

Syntax

```
BOOL GetGraphAxisInfo (
          short sectionN,
          short graphN,
          CRPEGraphAxisInfo *graphAxisInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to retrieve chart axis information.
graphN	Specifies for which chart within the section you want to retrieve chart axis information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphAxisInfo	Specifies a pointer to **CRPEGraphAxisInfo** (page 118), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGraphFontInfo

Use CRPEJob::GetGraphFontInfo to retrieve chart font information for the specified chart.

Syntax

```
BOOL GetGraphFontInfo (
           short sectionN,
           short graphN,
           WORD titleFontType,
           CRPEFontColorInfo *fontColourInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to retrieve the font information.
graphN	Specifies for which chart within the section you want to retrieve the font information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
titleFontType	Specifies the font type for the text. Use one of the PEP_GTF_XXX **Graph Text Font Constants** (page 169).
fontColourInfo	Specifies a pointer to **CRPEFontColorInfo** (page 116), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGraphOptionInfo

Use CRPEJob::GetGraphOptionInfo to obtain information about any of several chart options. These options include the minimum and maximum values that can appear on the chart, whether grid lines appear, whether risers are labeled, whether bar charts have horizontal or vertical bars, and whether a legend appears on the chart.

Syntax

```
BOOL GetGraphOptionInfo (
           short sectionN,
           short graphN,
           CRPEGraphOptionInfo * graphOptionInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to retrieve chart option information.
graphN	Specifies for which chart within the section you want to retrieve chart option information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphOptionInfo	Specifies a pointer to **CRPEGraphOptionInfo** (page 121).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGraphTextDefaultOption

Use CRPEJob::GetGraphTextDefaultOption to determine whether the graph text default option is enabled or disabled for the specified chart.

Syntax

```
BOOL GetGraphTextDefaultOption (
          short sectionN,
          short graphN,
          WORD titleType,
          BOOL FAR *useDefault );
```

Parameters

Parameter	*Description*
sectionN	Specifies the section of the report containing the chart for which you want to retrieve chart title text default information.
graphN	Specifies the 0-based index number of the chart within the section for which you want to retrieve the chart title text default information. Charts are numbered based on their order of insertion into the report.
titleType	Specifies the title type. Use one of the PE_GTT_XXX **Graph Title Type Constants** (page 168).
useDefault	Specifies the Boolean value indicating whether or not chart title defaults are enabled.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGraphTextInfo

Use CRPEJob::GetGraphTextInfo to determine what text appears with a chart. A chart 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 charts).

Syntax

```
BOOL GetGraphTextInfo (
          short sectionN,
          short graphN,
          WORD titleType,
          HANDLE FAR * title,
          short FAR * titleLength );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to get chart text information.
graphN	Specifies the chart for which you want to get graph text information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
titleType	Specifies the type of title. Use one of the PEP_GTT_XXX **Graph Title Type Constants** (page 168).
titleLength	Specifies a pointer to the handle of the title text.
titleLength	Specifies a pointer to the title length.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGraphTypeInfo

Use CRPEJob::GetGraphTypeInfo to retrieve chart type information for the specified chart in the specified section. There are many types of charts available for use in Seagate Crystal Reports. Use this method to find out what type of chart is being displayed in the report.

Syntax

```
BOOL GetGraphTypeInfo (
          short sectionN,
          short graphN,
          CRPEGraphTypeInfo *graphTypeInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to retrieve the type information.
graphN	Specifies for which chart within the section you want to retrieve the type information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphTypeInfo	Specifies a pointer to **CRPEGraphTypeInfo** (page 122), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGroupCondition

Use CRPEJob::GetGroupCondition to determine 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 84,* to change the group condition once it is known.

Syntax

```
BOOL GetGroupCondition (
          short sectionCode,
          CString &conditionField,
          short *condition,
          short *sortDirection );
```

Parameters

sectionCode	Specifies the code for the report section for which you want to get the grouping condition. Use one of the following constants.	
	Constant	***Description***
	PEP_GROUPHEADER	Sets the group condition for the Group Header section.
	PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.
conditionField	Reference to the name of the group summary field for which you want to get the grouping condition.	

condition	Specifies a pointer to 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:

Constant	***Description***
PEP_GC_CONDITIONMASK	Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the *condition* 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 *condition* 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 or type DateTime, the group condition value will be one of the following constants.

Constant	***Description***
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.

Constant	*Description*
PEP_GC_ANNUALLY	Triggers a grouping every time the date changes from one year to the next.

For a group condition field of type Time or DateTime, the group condition value will be one of the following constants.

Constant	*Description*
PEP_GC_BYSECOND	Triggers a grouping every second.
PEP_GC_BYMINUTE	Triggers a grouping every minute.
PEP_GC_BYHOUR	Triggers a grouping every hour.
PEP_GC_BYAMPM	Triggers a grouping at 0000 and 1200 hours.

For a group condition field of the type Boolean, the group condition value will be one of the following constants.

Constant	*Description*
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 sort-and-group-by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the sort-and-group-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 uses the following constants.

	Constant	*Description*
	PEP_GC_TYPEOTHER	Any field type other than Date or Boolean. The group condition portion of the *condition* 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.
	PEP_GC_TYPETIME	A Time field is used to create the group summary field.
sortDirection	Specifies a pointer to the sort direction for the group summary field. Use one of the PEP_SF_XXX **Sort Order Constants** (page 174).	

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGroupOptions

Use CRPEJob::GetGroupOptions to retrieve current settings for specified groups in a specified report.

Syntax

```
BOOL GetGroupOptions (
          short groupN,
          CRPEGroupOptions *groupOptions );
```

Parameters

groupN	Specifies the group number. The first level group is 0.
groupOptions	Specifies a pointer to **CRPEGroupOptions** (page 123).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetGroupSelectionFormula

Use CRPEJob::GetGroupSelectionFormula to retrieve the formula text for the group selection formula. This method is often used with *CRPEJob::SetGroupSelectionFormula, Page 86*, to identify and then change an existing group selection formula at print time (in response to a user selection).

Syntax

```
BOOL GetGroupSelectionFormula ( CString &formulaText );
```

Parameter

formulaText	Reference to the CString object containing the existing report group selection formula.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetJobHandle

Use CRPEJob::GetJobHandle to retrieve the job handle for the print job.

Syntax

```
short GetJobHandle ();
```

Returns

The print job handle for the CRPEJob object.

CRPEJob::GetJobStatus

Use CRPEJob::GetJobStatus to retrieve the current status of the print job.

Syntax

```
short GetJobStatus ( CRPEJobInfo *jobStatus );
```

Parameter

jobStatus	Specifies a pointer to **CRPEJobInfo** (page 127).

Returns

Returns one of the **Job Status Constants** (page 173), depending on the status of the current print job.

Remarks

You can use CRPEJob::GetJobStatus in a number of programming situations including the following.

- 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.
- To find out whether a job was canceled by the user after *CRPEJob::Start, Page 103*, was called.

CRPEJob::GetMargins

Use CRPEJob::GetMargins to retrieve the page margin settings for the specified report. Use this method to determine the current margin settings for a report. Use *CRPEJob::SetMargins, Page 87*, to change report margins.

Syntax

```
BOOL GetMargins (
           short *left,
           short *right,
           short *top,
           short *bottom );
```

Parameters

left	Specifies a pointer to the current setting of the left margin in twips.
right	Specifies a pointer to the current setting of the right margin in twips.
top	Specifies a pointer to the current setting of the top margin in twips.
bottom	Specifies a pointer to the current setting of the bottom margin in twips.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

PEP_SM_DEFAULT is used to indicate printer default margins.

CRPEJob::GetNDetailCopies

Use CRPEJob::GetNDetailCopies to determine the number of copies of each Details section in the report to be printed. Use this method to find out how many times each Details section of the report will be printed and use *CRPEJob::SetNDetailCopies, Page 88*, to change the number of times each Details section is printed.

Syntax

```
BOOL GetNDetailCopies ( short *nCopies );
```

Parameter

nCopies	Specifies a pointer to the current setting for the number of times the Details section of the report will be printed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNFormulas

Use CRPEJob::GetNFormulas to determine the number of formulas in the report. To retrieve the formula by number, use *CRPEJob::GetNthFormula, Page 48*.

Syntax

```
short GetNFormulas ();
```

Returns

- Returns the number of formulas in the report.
- Returns 0 if no formulas exist in the report.
- Returns -1 if an error occurs.

CRPEJob::GetNGroups

Use CRPEJob::GetNGroups to determine the number of group sections in the report.

Syntax

```
short GetNGroups ();
```

Returns

- Returns the number of group sections in the report.
- Returns -1 if an error occurs.

CRPEJob::GetNGroupSortFields

Use CRPEJob::GetNGroupSortFields to determine the number of group sort fields in the specified report. This method is typically used as one of a series of calls (GetNGroupSortFields, called once; *CRPEJob::GetNthGroupSortField, Page 49*, called as many times as needed to identify the correct group sort field;

and *CRPEJob::SetNthGroupSortField, Page 88*, called once when the correct sort field is identified). This sequence of calls can be used to identify and then change an existing group sort field and/or sort order in response to a user selection at print time.

Syntax

```
short GetNGroupSortFields ();
```

Returns

- Returns the number of group sort fields in the report.
- Returns 0 if there are no group sort fields.
- Returns -1 if an error occurs.

CRPEJob::GetNPages

Use CRPEJob::GetNPages to determine the number of pages in the report.

Syntax

```
short GetNPages ();
```

Returns

Returns the number of pages in the report.

CRPEJob::GetNParameterCurrentRanges

Use CRPEJob::GetNParameterCurrentRanges to determine the number of value ranges currently associated with the specified parameter field in the specified report.

Syntax

```
unsigned short GetNParameterCurrentRanges (
           const _TCHAR *parameterFieldName,
           const _TCHAR *reportName );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.

Returns

- Returns the number of value ranges for the parameter field.
- Returns -1 if the call fails.

CRPEJob::GetNParameterCurrentValues

Use CRPEJob::GetNParameterCurrentValues to determine the number of values currently stored in the specified parameter field of a report.

Syntax

```
unsigned short GetNParameterCurrentValues (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.

Returns

- Returns the number of values currently stored in the parameter field.
- Returns -1 if the call fails.

CRPEJob::GetNParameterDefaultValues

Use CRPEJob::GetNParameterDefaultValues to determine the number of default values for the specified parameter field in the specified report.

Syntax

```
short GetNParameterDefaultValues (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.

Returns

- Returns the number of default values for the parameter field.
- Returns 0 if there are no default values.
- Returns -1 if an error occurs.

CRPEJob::GetNParameterFields

Use CRPEJob::GetNParameterFields to determine the number of parameter fields used in the report.

Syntax

```
short GetNParameterFields ();
```

Returns

Returns the number of parameter fields used in the report.

Remarks

To determine whether a specific parameter field is a stored procedure, use *CRPEJob::GetNthParameterField, Page 52*.

CRPEJob::GetNSections

Use CRPEJob::GetNSections to determine the number of sections in the specified report.

Syntax

```
short GetNSections ();
```

Returns

Returns the number of sections in the report.

Remarks

By default, each report has five areas, each containing one of the following sections.

- Report Header
- Page Header
- Details
- Report Footer
- Page Footer

If this method were applied to a default report, the return value would be 5 (five). As you add groups to your report or you add sections to one or more areas, the number of sections in the report increases.

CRPEJob::GetNSectionsInArea

Use CRPEJob::GetNSectionsInArea to determine the number of sections in the specified area of the report.

Syntax

```
short GetNSectionsInArea ( short areaCode );
```

Parameter

areaCode	Specifies a pointer to the area code for which you want to retrieve the number of sections.

Returns

- Returns the number of sections in the specified area if the call is successful.
- Returns -1 if the call fails.

CRPEJob::GetNSortFields

Use CRPEJob::GetNSortFields to determine the number of sort fields in the report.

Syntax

```
short GetNSortFields ();
```

Returns

- Returns the number of sort fields defined in the report.
- Returns 0 if there are no sort fields in the report.
- Returns -1 if an error occurs.

CRPEJob::GetNSQLExpressions

Use CRPEJob::GetNSQLExpressions to determine the number of SQL expressions in the specified report.

Syntax

```
short GetNSQLExpressions ();
```

Returns

- Returns the number of SQL expressions in the Report.
- Returns 0 if there are no SQL expressions.
- Returns -1 if an error occurs.

CRPEJob::GetNSubreportsInSection

Use CRPEJob::GetNSubreportsInSection to determine the number of subreports in the specified section.

Syntax

```
short GetNSubreportsInSection ( short sectionCode );
```

Parameter

sectionCode	Specifies the section code of the section for which you want a subreport count. See *Working with section codes* in the *Seagate Crystal Reports Technical Reference Guide*.

Returns

- Returns the number of subreports in the report.
- Returns 0 if there are no subreports.
- Returns -1 if an error occurs.

CRPEJob::GetNTables

Use CRPEJob::GetNTables to determine the number of tables used in the report.

Syntax

```
short GetNTables ();
```

Returns

Returns the number of tables in the report.

CRPEJob::GetNthFormula

Use CRPEJob::GetNthFormula 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 by using *CRPEJob::SetFormula, Page 79*.

Syntax

```
BOOL GetNthFormula (
           short formulaN,
           CString &formulaName,
           CString &formulaText );
```

Parameters

formulaN	Specifies the number of the formula about which you want to retrieve information. The first formula added to your report is 0, the second is 1, etc.
formulaName	Reference to the CString containing the name of the formula specified.
formulaText	Reference to the CString containing the text of the formula specified.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthGroupSortField

Use CRPEJob::GetNthGroupSortField to determine the name of the group sort field and the sort direction (ascending or descending). This method is typically used as one of a series of calls (*CRPEJob::GetNGroupSortFields, Page 43*, called once; CRPEJob::GetNthGroupSortField, called as many times as needed to identify the correct group sort field; and *CRPEJob::SetNthGroupSortField, Page 88*, called once when the correct group sort field is identified). This series of methods can be used to identify and then change an existing group sort field and/or sort order in response to a user selection at print time.

Syntax

```
BOOL GetNthGroupSortField (
          short sortFieldN,
          CString &field,
          short *direction );
```

Parameters

sortFieldN	Specifies the number of the group sort field that you want to retrieve. The first group sort field is field 0. If the report has N sort fields, the method can be called with sortFieldN between 0 and N-1.
field	Reference to the CString object containing the name of the group sort field.
direction	Specifies a pointer to the value indicating the sort direction. Uses one of the PEP_SF_XXX **Sort Order Constants** (page 174).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthParameterCurrentRange

Use CRPEJob::GetNthParameterCurrentRange to retrieve the indicated value range from the specified parameter field in a report.

Syntax

```
BOOL GetNthParameterCurrentRange (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
```

```
CRPEValueInfo *rangeStart,
CRPEValueInfo *rangeEnd,
short *rangeInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
index	Specifies the index of the value range to be retrieved.
rangeStart	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the retrieved start range value. See Remarks below.
rangeEnd	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the retrieved end range value. See Remarks below.
rangeInfo	Specifies a pointer to the bit mapped value, which indicates whether the upper and/or lower bound of the range should be retrieved. Set rangeInfo to one of the following constants.

Constant	*Value*	*Description*
PEP_RI_INCLUDEUPPERBOUND	1	Returns *rangeStart.*
PEP_RI_INCLUDELOWERBOUND	2	Returns *rangeEnd.*
PEP_RI_INCLUDEUPPERBOUND OR PEP_RI_INCLUDELOWERBOUND	3	Returns *rangeStart* and *rangeEnd.*

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Set *rangeStart* to NULL to retrieve maximum value only; *rangeStart* must be non-NULL if *rangeEnd* is NULL.
- Set *rangeEnd* to NULL to retrieve minimum value only; *rangeEnd* must be non-NULL if *rangeStart* is NULL.
- If both *rangeStart* and *rangeEnd* are specified (that is, non-NULL), then the valueType member of **CRPEValueInfo** (page 152) for rangeStart and rangeEnd must be the same or an error code PEP_ERR_INCONSISTANTTYPES will be returned.

CRPEJob::GetNthParameterCurrentValue

Use CRPEJob::GetNthParameterCurrentValue to retrieve the indicated value from the specified parameter field of a specified report.

Syntax

```
BOOL GetNthParameterCurrentValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
          CRPEValueInfo *currentValue );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
index	Specifies the index number of the value to be retrieved.
currentValue	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the retrieved value.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthParameterDefaultValue

Use CRPEJob::GetNthParameterDefaultValue to retrieve the indicated default value for the specified parameter field in the specified report.

Syntax

```
BOOL GetNthParameterDefaultValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
          CRPEValueInfo *valueInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
index	The index of the default parameter field value to be retrieved.

valuInfo	Specifies a pointer to **CRPEValueInfo** (page 152), in which the default value will be returned.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthParameterField

Use CRPEJob::GetNthParameterField to retrieve 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 method is typically used as one of a series of calls (Returns -1 if an error occurs *CRPEJob::GetNParameterFields, Page 46*, called once; CRPEJob::GetNthParameterField, called as many times as needed to identify the correct parameter field; and *CRPEJob::SetNthParameterField, Page 89*, called once when the correct parameter field is identified). This series can be used in a Custom-Print Link to identify and then change an existing parameter field value in response to a user selection at print time.

Syntax

```
BOOL GetNthParameterField (
          short parameterN,
          CRPEParameterFieldInfo *parameterInfo );
```

Parameters

parameterN	Specifies the number of the parameter field about which you want to retrieve information.
parameterInfo	Specifies a pointer to **CRPEParameterFieldInfo** (page 131).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthParameterType

Use CRPEJob::GetNthParameterType to retrieve the type or origin of a specified parameter field.

Syntax

```
short GetNthParameterType ( short index );
```

Parameter

index	Specifies the index of the parameter field for which you want to retrieve the type.

Returns

- Returns one of the following values.

Constant	*Description*
PEP_PO_REPORT	Report
PEP_PO_STOREDPROC	Stored Procedure
PEP_PO_QUERY	Query

- Returns -1 if the index is not valid.

CRPEJob::GetNthParameterValue Description

Use CRPEJob::GetNthParameterValueDescription to retrieve the description of the default value set for a parameter.

Syntax

```
BOOL GetNthParameterValueDescription (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
          HANDLE *valueDesc,
          short *valueDescLength );
```

Parameters

parameterFieldName	Specifies a pointer to the parameterFieldName for which you want to retrieve the parameter value description.
reportName	Specifies a pointer to the report name.
index	Specifies the index.
valueDesc	Specifies a pointer to the handle of the value description to be retrieved.
valueDescLength	Specifies a pointer to the length of the value description.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthSortField

Use CRPEJob::GetNthSortField to retrieve the name of the sort field at the specified sort field position and the direction in which data in that field is sorted (ascending or descending).

Syntax

```
BOOL GetNthSortField (
          short sortFieldN,
          CString &field,
          short *direction );
```

Parameters

sortFieldN	Specifies the number of the sort field for which you want to retrieve information. The first sort field added to the report is field 0, the second is 1, etc.
field	Reference to the CString object containing the name of the sort field if the call is completed successfully.
direction	Specifies a pointer to the sort direction of the sort field if the call is completed successfully. Uses one of the PEP_SF_XXX **Sort Order Constants** (page 174).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthSQLExpression

Use CRPEJob::GetNthSQLExpression to retrieve a specified SQL expression.

Syntax

```
BOOL GetNthSQLExpression (
          short expressionN,
          CString &expressionName,
          CString &expressionText );
```

Parameters

expressionN	Index of the SQL expression that you want to retrieve.
expressionName	Reference to the CString object containing the SQL expression name.
expressionText	Reference to the CString object containing the SQL expression.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthSubreportInSection

Use CRPEJob::GetNthSubreportInSection to retrieve a handle that is required to retrieve the name of the subreport.

Syntax

```
DWORD GetNthSubreportInSection (
          short sectionCode,
          short subreportN );
```

Parameters

sectionCode	Specifies the code for the report section that contains the subreport. See *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.
subreportN	Specifies the 0-based number of the subreport in the specified section. The first report in the section will be 0, the second will be 1, etc.

Returns

- Returns a handle to the specified subreport.
- Returns 0 if there are no subreports in the specified section.
- Returns -1 if an error occurs.

CRPEJob::GetNthTableLocation

Use CRPEJob::GetNthTableLocation to retrieve table location information for the specified table in the report. This method is typically used with *CRPEJob::SetNthTableLocation, Page 91*, to identify the location of a table and then to change it.

Syntax

```
BOOL GetNthTableLocation (
          short tableN,
          CRPETableLocation *tableLocation );
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to retrieve location information.
tableLocation	Specifies a pointer to **CRPETableLocation** (page 148).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthTableLogOnInfo

Use CRPEJob::GetNthTableLogOnInfo to retrieve SQL connection information for the specified table.

Syntax

```
BOOL GetNthTableLogonInfo (
          short tableN,
          CRPELogOnInfo *logonInfo );
```

Parameters

tableN	Specifies the 0-based number of the table from which you want to get logon information.
logonInfo	Specifies a pointer to **CRPELogOnInfo** (page 128).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthTablePrivateInfo

Use CRPEJob::GetNthTablePrivateInfo to retrieve database information from **CRPETablePrivateInfo** (page 149).

Syntax

```
BOOL GetNthTablePrivateInfo (
          short tableN,
          CRPETablePrivateInfo *privateInfo );
```

Parameters

tableN	Specifies the 0-based number of the table from which you want to get private information.
privateInfo	Specifies a pointer to **CRPETablePrivateInfo** (page 149), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthTableSessionInfo

Use CRPEJob::GetNthTableSessionInfo to retrieve 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 table.

Syntax

```
BOOL GetNthTableSessionInfo (
          short tableN,
          CRPESessionInfo *sessionInfo );
```

Parameters

tableN	Specifies the 0-based table number indicating the MS Access table in the report for which you want to obtain the session information.
sessionInfo	Specifies a pointer to **CRPESessionInfo** (page 144).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNthTableType

Use CRPEJob::GetNthTableType to retrieve table type information for the specified table.

Syntax

```
BOOL GetNthTableType (
          short tableN,
          CRPETableType *tableType );
```

Parameters

tableN	Specifies the 0-based table number indicating the table in the report for which you want to determine the type.
tableType	Specifies a pointer to **CRPETableType** (page 150).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetParameterMinMaxValue

Use CRPEJob::GetParameterMinMaxValue to retrieve the minimum and/or maximum possible values for a specified parameter in the specified report.

Syntax

```
BOOL GetParameterMinMaxValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEValueInfo *valueMin,
          CRPEValueInfo *valueMax );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
valueMin	Specifies a pointer to **CRPEValueInfo** (page 152), which contains the minimum value.
valueMax	Specifies a pointer to **CRPEValueInfo** (page 152), which contains the maximum value.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Set valueMin to NULL to retrieve maximum value only; must be non-NULL if valueMax is NULL.
- Set valueMax to NULL to retrieve minimum value only; must be non-NULL if valueMin is NULL.

CRPEJob::GetParameterPickListOption

Use CRPEJob::GetParameterPickListOption to retrieve the parameter pick list options for a report. This method retrieves the values in **CRPEParameterPickListOption** (page 133).

Syntax

```
BOOL GetParameterPickListOption (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEParameterPickListOption *pickListOption );
```

Parameters

parameterFieldName	Specifies a pointer to the parameterFieldName for which you want to retrieve pick list options.
reportName	Specifies a pointer to the report name.
pickListOption	Specifies a pointer to **CRPEParameterPickListOption** (page 133), which will contain the information retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetParameterValueInfo

Use CRPEJob::GetParameterValueInfo to retrieve the CRPEParameterValueInfo structure associated with the specified parameter field in a report. This structure contains information about the values (editable, nullable field, multiple values, etc.) that can be stored in the specified field.

Syntax

```
BOOL GetParameterValueInfo (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEParamelterValueInfo *valueInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
valueInfo	Specifies a pointer to **CRPEParameterValueInfo** (page 134), which contains the retrieved value information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

To determine whether a parameter field is a stored procedure, use *CRPEJob::GetNthParameterField, Page 52*.

CRPEJob::GetPrintDate

Use CRPEJob::GetPrintDate to determine the print date (if any) that was specified with the report. Use this method to retrieve the print date and use *CRPEJob::SetPrintDate, Page 95*, to pass a new print date.

Syntax

```
BOOL GetPrintDate (
          short *year,
          short *month,
          short *day );
```

Parameters

year	Specifies a pointer to the year for the current print date.
month	Specifies a pointer to the month for the current print date.
day	Specifies a pointer to the day for the current print date.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetPrintOptions

Use CRPEJob::GetPrintOptions to retrieve the print options specified for the report (the options that are set in the Print Setup common dialog box). Use this method to retrieve the print options from the report in order to update them and use *CRPEJob::SetPrintOptions, Page 96*, to pass new print options back to the structure.

Syntax

```
BOOL GetPrintOptions ( CRPEPrintOptions *options );
```

Parameter

options	Specifies a pointer to **CRPEPrintOptions** (page 135).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetReportOptions

Use CRPEJob::GetReportOptions to retrieve various options specified for the report from the Seagate Crystal Reports Designer File | Report Options menu **CRPEReportOptions** (page 138). Use this method to retrieve print options from the report and use *CRPEJob::SetReportOptions, Page 96*, to pass new option settings back to the structure.

Syntax

```
BOOL GetReportOptions ( CRPEReportOptions *reportOptions );
```

Parameter

reportOptions	Specifies a pointer to the structure, **CRPEReportOptions** (page 138), containing the report options information to be retrieved.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetReportSummaryInfo

Use CRPEJob::GetReportSummaryInfo to retrieve summary information about a report such as the report title, author, and comments.

Syntax

```
BOOL GetReportSummaryInfo ( CRPEReportSummaryInfo *summaryInfo );
```

Parameter

summaryInfo	Specifies a pointer to **CRPEReportSummaryInfo** (page 140).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetReportTitle

Use CRPEJob::GetReportTitle to retrieve the report title for the print job.

Syntax

```
BOOL GetReportTitle ( CString &title );
```

Parameter

title	Refers to the CString object containing the title of the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSectionCode

Use CRPEJob::GetSectionCode to retrieve the section code for the specified section. A section code indicates the section type (Page Header, Details, etc.). 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.

Syntax

```
short GetSectionCode ( short sectionN );
```

Parameter

sectionN	Specifies the number of the section for which you want to retrieve the section code.

Returns

- Returns the section code for the specified section.
- Returns 0 if the call fails.

CRPEJob::GetSectionFormat

Use CRPEJob::GetSectionFormat to retrieve the section format settings for a selected section in the specified report and supply the information by assigning values to the members of the structure, **CRPESectionOptions** (page 142). Use this method in order to edit and update the section formats and use *CRPEJob::SetSectionFormat, Page 97*, to pass information back to CRPESectionOptions.

Syntax

```
BOOL GetSectionFormat (
          short sectionCode,
          CRPESectionOptions *options );
```

Parameters

sectionCode	Specifies the code for the report section for which you want to obtain format information. Use one of the PEP_XXX **Section Codes Constants** (page 174).
options	Specifies a pointer to **CRPESectionOptions** (page 142).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSectionFormatFormula

Use CRPEJob::GetSectionFormatFormula to retrieve the text of a conditional section format formula as a string handle. Use this method to update the conditional section format formula and pass the changes back using *CRPEJob::SetSectionFormatFormula, Page 98*.

Syntax

```
BOOL GetSectionFormatFormula (
          short areaCode,
          short sectionCode,
          short formulaName,
          CString &formulaText );
```

Parameters

areaCode	Specifies the code for the report area for which you want to set formatting options. For information on area codes, see *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.
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 *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.
formulaName	Specifies the name of the PEP_FFN_XXX **Area/Section Format Formula Constants** (page 157).
formulaText	Refers to the Cstring object passed with the specified formula text when CRPEJob::GetSectionFormatFormula has completed successfully.

Returns

- TRUE if the call is successful.
- FALSE 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 *CRPEngine::GetHandleString, Page 6*, to fill the buffer with the actual text of the formula.

CRPEJob::GetSectionHeight

Use CRPEJob::GetSectionHeight to retrieve the section height information for the specified section. Use this method to retrieve the section height, and use *CRPEJob::SetSectionHeight, Page 99*, to pass new section height information.

Syntax

```
BOOL GetSectionHeight (
           short sectionCode,
           short *height );
```

Parameters

sectionCode	Specifies the code for the report sections for which you want to retrieve information.
height	Specifies a pointer to the section height (in twips) returned by this call.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSelectedPrinter

Use CRPEJob::GetSelectedPrinter to determine which printer has been specified with the report. If a non-default printer is specified in the report, CRPEJob::GetSelectedPrinter retrieves information about that printer. This can be used, for example, to determine if the user has access to the printer specified in the report and to choose another printer if necessary.

Syntax

```
BOOL GetSelectedPrinter (
           CString &driverName,
           CString &printerName,
           CString &portName,
           DEVMODE **mode );
```

Parameters

driverName	Reference to the CString object containing the name of the printer driver for the currently selected printer in the report.
printerName	Reference to the CString object containing the name of the printer currently selected in the report.
portName	Reference to the CString object containing the name of the port to which the currently selected printer is attached (for example, "LPT1").
mode	Specifies a pointer to a pointer to the Windows API structure DEVMODE that contains information on the currently selected printer, if the CRPEJob::GetSelectedPrinter method is completed successfully. For more information on the DEVMODE structure, see **DEVMODE** in the *Seagate Crystal Reports Technical Reference Guide*.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSelectionFormula

Use CRPEJob::GetSelectionFormula to retrieve the formula text for the record selection formula. CRPEJob::GetSelectionFormula is often used with *CRPEJob::SetSelectionFormula, Page 99*, to identify and then change an existing selection formula in response to a user selection at print time.

Syntax

```
BOOL GetSelectionFormula (
           CString &formulaText );
```

Parameter

formulaText	Reference to the CString object containing the existing selection formula for the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSQLExpression

Use CRPEJob::GetSQLExpression to retrieve the text of a specific SQL expression, given an expression name.

Syntax

```
BOOL GetSQLExpression (
          const _TCHAR *expressionName,
          CString &expressionText );
```

Parameters

expressionName	Specifies a pointer to the expression name.
expressionText	Reference to the CString that contains the expression text.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSQLQuery

Use CRPEJob::GetSQLQuery to retrieve the SQL query that will be sent to the database server. This method can be used with *CRPEJob::SetSQLQuery, Page 100,* to retrieve and then change the SQL query for the report.

Syntax

```
BOOL GetSQLQuery ( CString &query );
```

Parameter

query	Reference to the CString containing the text of the SQL query being sent to the server if the call is completed successfully.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetSubreportInfo

Use CRPEJob::GetSubreportInfo to retrieve information about the specified subreport.

Syntax

```
BOOL GetSubreportInfo (
          DWORD subreportHandle,
          CRPESubreportInfo *subreportInfo );
```

Parameters

subreportHandle	Specifies the handle of the subreport about which you want to retrieve information.
subreportInfo	Specifies a pointer to **CRPESubreportInfo** (page 147).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetTrackCursorInfo

Use CRPEJob::GetTrackCursorInfo to retrieve track cursor details for the preview window.

Syntax

```
BOOL GetTrackCursorInfo ( CRPETrackCursorInfo *cursorInfo );
```

Parameter

cursorInfo	Specifies a pointer to **CRPETrackCursorInfo** (page 150).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetWindowHandle

Use CRPEJob::GetWindowHandle to retrieve 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.

Syntax

```
HWND GetWindowHandle ();
```

Returns

The Microsoft Windows Handle for the preview window.

CRPEJob::GetWindowOptions

Use CRPEJob::GetWindowOptions to retrieve the options associated with a specified window object. This information can then be changed and sent back to **CRPEWindowOptions** (page 153) by using *CRPEJob::SetWindowOptions, Page 101*. Use this method to customize the features of the window object.

Syntax

```
BOOL GetWindowOptions ( CRPEWindowOptions *windowOptions );
```

Parameter

windowOptions	Specifies a pointer to **CRPEWindowOptions** (page 153), which contains the information that you want to retrieve.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::HasSavedData

Use CRPEJob::HasSavedData to determine whether a report has saved data. This method can be used to determine whether it is necessary to refresh the data before printing the report.

Syntax

```
BOOL HasSavedData ( BOOL *hasSavedData );
```

Parameter

hasSavedData	Specifies a pointer to a memory address that indicates whether there is data saved with the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::IsJobFinished

Use CRPEJob::IsJobFinished to retrieve a Boolean value that indicates whether processing has finished for the print job. You can use this method any time that you have a call that is contingent on a print job being finished.

Syntax

```
BOOL IsJobFinished ();
```

Returns

- TRUE if processing has finished.
- FALSE if the job is in progress.

CRPEJob::NextWindowMagnification

Use CRPEJob::NextWindowMagnification to switch to the next preview window magnification in order. Use this method to cycle through the three levels (Full page, Fit one side, and Fit both sides) of preview window magnification, whenever the report has been printed to a preview window.

Syntax

```
BOOL NextWindowMagnification ();
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::OpenSubreportJob

Use CRPEJob::OpenSubreportJob to open the named subreport and return a pointer to a CRPEJob object for that subreport. The CRPEJob object can be used to make subsequent changes to the subreport.

Syntax

```
CRPEJob *OpenSubreportJob ( const _TCHAR *subreportName );
```

Parameter

subreportName	Specifies a pointer to the name of the subreport that you want to access. This name is retrieved using CRPEJob::GetSubreportInfo, Page 66.

Returns

A pointer to a **class CRPEJob : public CObject** (page 12), object for the subreport.

CRPEJob::OutputToPrinter

Use CRPEJob::OutputToPrinter to set 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. Use *CRPEJob::Start, Page 103* to print the report.

Syntax

```
BOOL OutputToPrinter ( short nCopies = 1 );
```

Parameter

nCopies	Specifies how many copies of the report are to be printed. Default is 1 copy.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::OutputToWindow

Use CRPEJob::OutputToWindow to set the output of the print job to the preview window that will have the specified attributes. It does not print the report, but specifies that when the report is printed, it will also appear in a preview window. To print the report, use *CRPEJob::Start, Page 103*.

Syntax

For an SDI application

```
BOOL OutputToWindow (
          const _TCHAR *title,
          int left,
          int top,
          int width,
          int height,
          int style,
          CWnd *parentWindow );
```

For an MDI application

```
BOOL OutputToWindow (
          const _TCHAR *title,
          int left,
          int top,
          int width,
          int height,
          int style,
          CMDIFrameWnd *parentWindow );
```

Parameters

title	Specifies a pointer to the title that 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

This method is overloaded to handle both SDI and MDI applications.

CRPEJob::PrintControlsShowing

Use CRPEJob::PrintControlsShowing to determine if the print controls will be displayed in the preview window. Use *CRPEJob::ShowPrintControls, Page 102*, to specify whether or not the print controls will appear in the preview window.

Syntax

```
BOOL PrintControlsShowing ( BOOL *controlsShowing );
```

Parameter

controlsShowing	Specifies a pointer to a TRUE value if the print controls will be shown in the preview window and a FALSE value if they will be hidden.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::PrintWindow

Use CRPEJob::PrintWindow to send 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 send the report to the printer in response to a user event (button click, menu command, etc.).

Syntax

```
BOOL PrintWindow ();
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ReimportSubreport

Use CRPEJob::ReimportSubreport to reimport a subreport into the specified main report.

Syntax

```
BOOL ReimportSubreport ( DWORD subreportHandle,
          BOOL FAR *linkChanged,
          BOOL FAR *reimported );
```

Parameters

Parameter	*Description*
subreportHandle	Specifies the handle of the subreport that you want to reimport.
linkChanged	Specifies a pointer to a Boolean value indicating whether or not the link has changed. See Remarks below.
reimported	Specifies a pointer to a Boolean value indicating whether or not the subreport has been reimported. See Remarks below.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Parameter linkChanged
  - — will be set to FALSE if the subreport is reloaded and the links are fixed.
  - — will be set to TRUE if the subreport is reloaded but missing links.
- Parameter reimported
  - — will be set to FALSE if the subreport is up-to-date, or if the reimport failed due to an invalid path or other error.
  - — will be set to TRUE if the subreport was reloaded with links fixed or missing.

CRPEJob::SelectPrinter

Use CRPEJob::SelectPrinter to specify 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.

Syntax

```
BOOL SelectPrinter (
          const _TCHAR *driverName,
          const _TCHAR *printerName,
          const _TCHAR *portName,
          const DEVMODE *mode = 0 );
```

Parameters

driverName	Specifies a pointer to the name of the printer driver for the printer being selected.
printerName	Specifies a pointer to the name of the printer being selected (as indicated in the Printers Control Panel).
portName	Specifies a pointer to the name of the port to which the printer is attached (for example, "LPT1").
mode	Specifies a pointer to a Windows API DEVMODE structure. The default implementation of CRPEJob::SelectPrinter ignores this parameter. For more information, see **DEVMODE** in the *Seagate Crystal Reports Technical Reference Guide*.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetAllowPromptDialog

Use CRPEJob::SetAllowPromptDialog to specify whether prompting for parameter values is allowed during printing.

Syntax

```
BOOL SetAllowPromptDialog ( BOOL showPromptDialog );
```

Parameter

showPromptDialog	If *showPromptDialog is set to* TRUE, then prompting is allowed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetAreaFormat

Use CRPEJob::SetAreaFormat to set format information for the specified area.

Syntax

```
BOOL SetAreaFormat (
          short areaCode,
          CRPESectionOptions *options );
```

Parameters

areaCode	Specifies the area code for the area for which you want to set format information.
options	Specifies a pointer to **CRPESectionOptions** (page 142), which will contain the new area format information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetAreaFormatFormula

Use CRPEJob::SetAreaFormatFormula to change the specified area format formula to the formula string that you supply as a parameter. This method will change only the text of a formula that already exists in the report; you cannot 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.

Syntax

```
BOOL SetAreaFormatFormula (
          short areaCode,
          short formulaName,
          CString formulaText );
```

Parameters

areaCode	Specifies the code for the report area for which you want to set formatting options.
formulaName	Specifies the name of the formatting formula for which you want to supply a new string. Use one of the PEP_FFN_XXX **Area/Section Format Formula Constants** (page 157).

formulaText	The Cstring object passed here is loaded with the specified formula text when CRPEJob::`SetAreaFormatFormula` completes successfully.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- This method should be called before *CRPEJob::Start, Page 103* or the results may be inconsistent or unexpected.
- Not all parameters apply to all areas.

CRPEJob::SetDialogParentWindow

Use CRPEJob::SetDialogParentWindow to specify the handle for the parent window of a preview window that is an MDI child.

Syntax

```
BOOL SetDialogParentWindow ( CWnd *parentWindow );
```

Parameter

parentWindow	Specifies 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetEventCallback

Use CRPEJob::SetEventCallback to set event callback information for a job. The Seagate Crystal Reports Engine can determine when certain events have happened or are about to happen and provide notice so that appropriate response can be made.

Syntax

```
BOOL SetEventCallback (
          BOOL ( CALLBACK * callbackProc )
                       ( short eventID, void *param, void *userData ),
          void *userData );
```

Parameters

callbackProc	The CALLBACK procedure that will handle your Seagate Crystal Report Engine events. This should be a pointer to a standard Windows CALLBACK procedure. Refer to the Microsoft Windows SDK for information on creating CALLBACK procedures.
userData	Specifies a pointer to the information that you want to pass to the Event handling callback procedure. The pointer will be available in the user data member of callbackProc. This value can be 0.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

See the list of **Event Id Constants** (page 163). Also, for additional information, see the more detailed listing which follows *PESetEventCallback* in the *Seagate Crystal Reports Technical Reference Guide*.

CRPEJob::SetFieldMappingType

Use CRPEJob::SetFieldMappingType to set the field mapping type code.

Syntax

```
BOOL SetFieldMappingType ( WORD mappingType );
```

Parameter

mappingType	Specifies the field mapping type code. Use one of the following constants.	
	Constant	***Description***
	PEP_FM_AUTO_FLD_MAP	Automatic field name mapping.
	PEP_FM_CRPEP_PROMPT_FLD_MAP	CRPE provides dialog box to map field manually.
	PEP_FM_EVENT_DEFINED_FLD_MAP	CRPE provides list of fields in the report and the new database.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetFont

Use CRPEJob::SetFont to set the font and font characteristics for the specified section. Use it any time that you want to change a default font at runtime in response to user input, or to specify a built-in printer font.

Syntax

```
BOOL SetFont (
          short sectionCode,
          short scopeCode,
          const _TCHAR *faceName,
          short fontFamily,
          short fontPitch,
          short charSet,
          short pointSize,
          short isItalic,
          short isUnderlined,
          short isStruckOut,
          short weight );
```

Parameters

sectionCode	Specifies the section of the report for which you want to set the font. Use one of the PEP_XXX **Section Codes Constants** (page 174).	
scopeCode	Specifies whether the font selected is to apply to fields, to text, or to both. To specify both, use the bitwise "Or" operator (|). Use one of the following constants.	
	Constant	***Description***
	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.
faceName	Specifies a pointer to the actual face name of the font that 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 that you want to use. Use one of the following constants.	
	Constant	***Description***
	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.
fontPitch	Specifies the font pitch that you want to use. Use one of the following constants.	
	Constant	***Description***
	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.
charSet	Specifies the character set that you want to use. Use one of the following constants.	
	Constant	
	ANSI_CHARSET	
	SYMBOL_CHARSET	
	HANGEFUL_CHARSET	
	OEM_CHARSET	
	DEFAULT_CHARSET	
	SHIFTJIS_CHARSET	
	CHINESEBIG5_CHARSET	
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 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. Use one of the following constants.	
	Constant	
	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

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetFormula

Use CRPEJob::SetFormula to set the formula text for the specified formula. CRPEJob::SetFormula is often used with *CRPEJob::GetFormula, Page 32*, to identify and then change an existing formula in response to a user selection at print time.

Syntax

```
BOOL SetFormula (
          const _TCHAR *formulaName,
          const _TCHAR *formulaText );
```

Parameters

formulaName	Specifies a pointer to the name of the formula for which you want to assign new formula text.
formulaText	Specifies a pointer to the actual formula text that replaces the existing formula string.

Returns

- TRUE if the call is successful.
- FALSE if the named formula does not exist, if there is an error in the formula, or if the call fails.

CRPEJob::SetFormulaSyntax

Use CRPEJob::SetFormulaSyntax to set the syntax to use in the next (and all successive) formula API calls.

Syntax

```
BOOL SetFormulaSyntax ( CRPEFormulaSyntax *formulaSyntax );
```

Parameter

Parameter	*Description*
formulaSyntax	Specifies a pointer to **CRPEFormulaSyntax** (page 117) which will contain the information that you set. See Remarks below.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

For running total condition formula:

- formulaSyntax[0] is the syntax for the evalFormula, and
- formulaSyntax[1] is the syntax for the resetFormula.

CRPEJob::SetGraphAxisInfo

Use CRPEJob::SetGraphAxisInfo to set chart axis information for the specified chart.

Syntax

```
BOOL SetGraphAxisInfo (
          short sectionN,
          short graphN,
          CRPEGraphAxisInfo *graphAxisInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to set chart axis information.
graphN	Specifies for which chart within the section you want to set chart axis information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphAxisInfo	Specifies a pointer to **CRPEGraphAxisInfo** (page 118), which will contain the new information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGraphFontInfo

Use CRPEJob::SetGraphFontInfo to set graph font information for the specified chart.

Syntax

```
BOOL SetGraphFontInfo (
           short sectionN,
           short graphN,
           WORD titleFontType,
           CRPEFontColorInfo *fontColourInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to set the font information.
graphN	Specifies for which chart within the section you want to set the font information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
titleFontType	Specifies the font type for the text. Use one of the PEP_GTF_XXX **Graph Text Font Constants** (page 169).
fontColourInfo	Specifies a pointer to **CRPEFontColorInfo** (page 116), which will contain the new information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGraphOptionInfo

Use CRPEJob::SetGraphOptionInfo to change any of several chart options. These options include the minimum and maximum values that can appear on the chart, whether grid lines appear, whether risers are labeled, whether bar graphs have horizontal or vertical bars, and whether a legend appears on the chart.

Syntax

```
BOOL SetGraphOptionInfo (
           short sectionN,
           short graphN,
           CRPEGraphOptionInfo *graphOptionInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to set chart options.
graphN	Specifies for which chart within the section you want to set chart options. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphOptionInfo	Specifies a pointer to **CRPEGraphOptionInfo** (page 121).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGraphTextDefaultOption

Use CRPEJob::SetGraphTextDefaultOption to enable or disable chart title defaults.

Syntax

```
BOOL SetGraphTextDefaultOption (
          short sectionN,
          short graphN,
          WORD titleType,
          BOOL useDefault );
```

Parameters

Parameter	*Description*
sectionN	Specifies the section of the report containing the chart for which you want to set chart title text default information.
graphN	Specifies the 0-based index number of the chart within the section for which you want to set the chart title text default information. Charts are numbered based on their order of insertion into the report.
titleType	Specifies the title type. Use one of the PE_GTT_XXX **Graph Title Type Constants** (page 168).
useDefault	Specifies the Boolean value indicating whether or not chart title defaults are enabled.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGraphTextInfo

Use CRPEJob::SetGraphTextInfo to set or change the text that appears with a chart. A chart 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).

Syntax

```
BOOL SetGraphTextInfo (
          short sectionN,
          short graphN,
          WORD titleType,
          LPCSTR title );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to set the text information.
graphN	Specifies for which chart within the section you want to set the text information. This value is 0-based. Charts are numbered based on their order of insertion into the report.
titleType	Specifies the title type. Use one of the PEP_GTT_XXX **Graph Title Type Constants** (page 168).
title	Specifies a pointer to the CString object containing the title text.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGraphTypeInfo

Use CRPEJob::SetGraphTypeInfo to set the chart type information (based on one of the available chart types) for the specified chart in the specified section. Use this method to change the type of chart that is displayed in a report.

Syntax

```
BOOL SetGraphTypeInfo (
   short sectionN,
   short graphN,
   CRPEGraphTypeInfo *graphTypeInfo );
```

Parameters

sectionN	Specifies the section of the report containing the chart for which you want to set the type information.
graphN	Specifies for which chart within the section you want to set the type. This value is 0-based. Charts are numbered based on their order of insertion into the report.
graphTypeInfo	Specifies a pointer to **CRPEGraphTypeInfo** (page 122), which will contain the new information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGroupCondition

Use CRPEJob::SetGroupCondition to set the condition of the grouping for the specified group section. This method can only replace the group condition for an existing group. It cannot create a new group. Use this method whenever you want to change the grouping at print time (for example, to print a report grouped in several different ways).

Syntax

```
BOOL SetGroupCondition (
          short sectionCode,
          const _TCHAR *conditionField,
          short condition,
          short sortDirection );
```

Parameters

sectionCode	Specifies the code for the group section for which you want to set the group condition. Use one of the following constants.	
	Constant	***Description***
	PEP_GROUPHEADER	Sets the group condition for the Group Header section.
	PEP_GROUPFOOTER	Sets the group condition for the Group Footer section.
conditionField	Specifies a pointer to 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. Use one of the following constants for Date and Boolean field types.

Constant (Date fields only)	***Description***
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.
Constant (Boolean fields only)	***Description***
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 sort-and-group-by field value is Yes.
PEP_GC_EVERYNO	Triggers a grouping every time the sort-and-group-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.
sortDirection	Specifies one of the PEP_SF_XXX **Sort Order Constants** (page 174).	

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetGroupOptions

Use CRPEJob::SetGroupOptions to set various grouping options for the specified group. This method sets the formula text for the group selection formula. CRPEJob::SetGroupSelectionFormula is often used with *CRPEJob::GetGroupSelectionFormula, Page 41*, to identify and then change an existing group selection formula in response to a user selection at print time.

Syntax

```
BOOL SetGroupOptions (
          short groupN,
          CRPEGroupOptions *groupOptions );
```

Parameters

groupN	Specifies the 0-based group level number.
groupOptions	Specifies a pointer to **CRPEGroupOptions** (page 123).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- This method should be called before *CRPEJob::Start, Page 103*, or the results may be inconsistent or unexpected.
- If you are using *CRPEJob::SetGroupOptions, Page 86*, 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.

CRPEJob::SetGroupSelectionFormula

Use CRPEJob::SetGroupSelectionFormula to set the formula text for the group selection formula. This method is often used with *CRPEJob::GetGroupSelectionFormula, Page 41*, to identify and then change an existing group selection formula in response to a user selection at print time.

Syntax

```
BOOL SetGroupSelectionFormula ( const _TCHAR *formulaText );
```

Parameter

formulaText	Specifies a pointer to the new group selection formula to be assigned to the report.

Returns

- TRUE if the call is successful.
- FALSE if there is an error in the formula or if the call fails.

CRPEJob::SetMargins

Use CRPEJob::SetMargins to set the page margins for the print job. Use this method any time that you want to allow the user to change margins.

Syntax

```
BOOL SetMargins (
           short left,
           short right,
           short top,
           short bottom );
```

Parameters

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

For any of the parameters, PM_SM_DEFAULT can be used to specify that the report use default printer margins.

CRPEJob::SetNDetailCopies

Use CRPEJob::SetNDetailCopies to set 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.

Syntax

```
BOOL SetNDetailCopies ( short nCopies );
```

Parameter

nCopies	Specifies the number of copies of the Details section of the report to be printed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthGroupSortField

Use CRPEJob::SetNthGroupSortField to set the name of the group sort field and the sort direction (ascending or descending). This method is typically used as one of a series of calls (*CRPEJob::GetNGroupSortFields, Page 43*, called once; *CRPEJob::GetNthGroupSortField, Page 49*, called as many times as needed to identify the correct group sort field; and CRPEJob::SetNthGroupSortField called once when the correct group sort field is identified). This series of methods can be used to identify and then change an existing group sort field and/or sort order in response to a user selection at print time. This method does not create a new group, but will sort an existing group summary field.

Syntax

```
BOOL SetNthGroupSortField (
          short sortFieldN,
          const _TCHAR *field,
          short direction );
```

Parameters

sortFieldN	Specifies the number of the group sort field that 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 a pointer to the name of the group field to be sorted.
direction	Specifies the sort direction. Uses one of the PEP_SF_XXX **Sort Order Constants** (page 174).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthParameterDefaultValue

Use CRPEJob::SetNthParameterDefaultValue to set the indicated default value for the specified parameter field in the specified report.

Syntax

```
BOOL SetNthParameterDefaultValue (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
          CRPEValueInfo *valueInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
index	Specifies the index number of the default value that you want to set.
valueInfo	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the new value.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthParameterField

Use CRPEJob::SetNthParameterField to set a value for the specified parameter field. This method is typically used as one of a series of calls (*CRPEJob::GetNParameterFields, Page 46*, called once; *CRPEJob::GetNthParameterField, Page 52*, called as many times as needed to identify the correct parameter field; and CRPEJob::SetNthParameterField, called once when the correct parameter field is identified). This series can be used in a Custom-Print Link to identify and then change an existing parameter field value in response to a user selection at print time.

Syntax

```
BOOL SetNthParameterField (
          short parameterN,
          const CRPEParameterFieldInfo *parameterInfo );
```

Parameters

parameterN	Specifies the number of the parameter field for which you want to set information.
parameterInfo	Specifies a pointer to **CRPEParameterFieldInfo** (page 131).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthParameterValueDescription

Use CRPEJob::SetNthParameterValueDescription to set the description of the default value for a parameter.

Syntax

```
BOOL SetNthParameterValueDescription (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          short index,
          _TCHAR *valueDesc );
```

Parameters

parameterFieldName	Specifies a pointer to the parameterFieldName for which you want to set the parameter value description.
reportName	Specifies a pointer to the report name.
index	Specifies the index.
valueDesc	Specifies a pointer to the value description to be set.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthSortField

Use CRPEJob::SetNthSortField to sort report data according to the specified field and direction.

Syntax

```
BOOL SetNthSortField (
          short sortFieldN,
          const _TCHAR *field,
          short direction );
```

Parameters

sortFieldN	Specifies the number of the sort field that 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 a pointer to the name of the field to be sorted.
direction	Specifies the sort direction. Use one of the PEP_SF_XXX **Sort Order Constants** (page 174).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthTableLocation

Use CRPEJob::SetNthTableLocation to set the table location information for the specified table in the report. This method is typically combined with *CRPEJob::GetNthTableLocation, Page 55*, to identify the location of a table and CRPEJob::SetNthTableLocation to change it.

Syntax

```
BOOL SetNthTableLocation (
          short tableN,
          const CRPETableLocation *tableLocation );
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to set a new location.
tableLocation	Specifies a pointer to **CRPETableLocation** (page 148).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthTableLogOnInfo

Use CRPEJob::SetNthTableLogOnInfo to set the 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.

Syntax

```
BOOL SetNthTableLogonInfo (
          short tableN,
          const CRPELogOnInfo *logonInfo,
          BOOL propagate );
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to set logon information.
logonInfo	Specifies a pointer to **CRPELogOnInfo** (page 128).
propagate	Specify TRUE or FALSE to indicate whether the logon information should be used for opening all tables being used in the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthTablePrivateInfo

Use CRPEJob::SetNthTablePrivateInfo to set database information in **CRPETablePrivateInfo** (page 149).

Syntax

```
BOOL SetNthTablePrivateInfo (
          short tableN,
          CRPETablePrivateInfo *privateInfo );
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to set private information.
privateInfo	Specifies a pointer to **CRPETablePrivateInfo** (page 149), which will contain the new information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthTableSessionInfo

Use CRPEJob::SetNthTableSessionInfo to set the 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.

Syntax

```
BOOL SetNthTableSessionInfo (
          short tableN,
          const CRPESessionInfo *sessionInfo,
          BOOL propagate );
```

Parameters

tableN	Specifies the 0-based table number indicating for which MS Access table in the report the session is being opened.
sessionInfo	Specifies a pointer to the structure, **CRPESessionInfo** (page 144).
propagate	Specify TRUE or FALSE to indicate whether the session information should be used for opening all tables being used in the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetParameterMinMaxValue

Use CRPEJob::SetParameterMaxMinValue to set the minimum and/or maximum possible values for a specified parameter in a report.

Syntax

```
BOOL SetParameterMinMaxValue (
                  const _TCHAR *parameterFieldName,
                  const _TCHAR *reportName,
                  CRPEValueInfo *valueMin,
                  CRPEValueInfo *valueMax );
```

Parameters

parameterfieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
valueMin	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the minimum value. See Remarks below.

valueMax	Specifies a pointer to **CRPEValueInfo** (page 152), which will contain the maximum value. See Remarks below.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Set valueMin to NULL if specifying maximum value only; valueMin must be non-NULL if valueMax is NULL.
- Set valueMax to NULL if specifying minimum value only; valueMax must be non-NULL if valueMin is NULL.
- If both valueMin and valueMax are specified (that is, non-NULL), then the valueType field of both members must be the same or an error code PEP_ERR_INCONSISTANTTYPES will be returned.

CRPEJob::SetParameterPickListOption

Use CRPEJob::SetParameterPickListOption to set the parameter pick list options for a report. This method sets the values in **CRPEParameterPickListOption** (page 133).

Syntax

```
BOOL SetParameterPickListOption (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEParameterPickListOption *pickListOption );
```

Parameters

parameterFieldName	Specifies a pointer to the parameterFieldName for which you want to set pick list options.
reportName	Specifies a pointer to the report name.
pickListOption	Specifies a pointer to **CRPEParameterPickListOption** (page 133), which will contain the new information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetParameterValueInfo

Use CRPEJob::SetParameterValueInfo to set the information about the values (editable, nullable field, multiple values, etc.) that can be stored in a specified parameter field.

Syntax

```
BOOL SetParameterValueInfo (
          const _TCHAR *parameterFieldName,
          const _TCHAR *reportName,
          CRPEParameterValueInfo *valueInfo );
```

Parameters

parameterFieldName	Specifies a pointer to the parameter field name.
reportName	Specifies a pointer to the report name.
valueInfo	Specifies a pointer to **CRPEParameterValueInfo** (page 134), which will contain the new value information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetPrintDate

Use CRPEJob::SetPrintDate to set only the date that appears in the Print Date Field of 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 of the report (for example, use this method to post-date a report when it is printed before the day it is distributed).

Syntax

```
BOOL SetPrintDate (
          short year,
          short month,
          short day );
```

Parameters

year	Specifies the year that will appear in the Print Date Field.
month	Specifies the month that will appear in the Print Date Field.
day	Specifies the day that will appear in the Print Date Field.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetPrintOptions

Use CRPEJob::SetPrintOptions to set the print characteristics of a print job. Use this method any time that you want to set the starting page number, the ending page number, the number of report copies, and/or collation instructions in response to user specifications at runtime.

Syntax

```
BOOL SetPrintOptions ( const CRPEPrintOptions *options );
```

Parameter

options	Specifies a pointer to **CRPEJob::SetPrintOptions** (page 96), which will contain the information that you want to set.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetReportOptions

Use CRPEJob::SetReportOptions to set the report options associated with a specified print job. This information is specified from the Seagate Crystal Reports Designer File | Report Options menu or by *CRPEJob::GetReportOptions, Page 61*.

Syntax

```
BOOL SetReportOptions ( CRPEReportOptions *reportOptions );
```

Parameter

reportOptions	Specifies a pointer to **CRPEReportOptions** (page 138), which will contain the new report options information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetReportSummaryInfo

Use CRPEJob::SetReportSummaryInfo to set report summary information. Report summary information corresponds to the Summary Info dialog box in Seagate Crystal Reports.

Syntax

```
BOOL SetReportSummaryInfo ( CRPEReportSummaryInfo *summaryInfo );
```

Parameter

summaryInfo	Specifies a pointer to **CRPEReportSummaryInfo** (page 140), which will contain the new report summary information.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetReportTitle

Use CRPEJob::SetReportTitle to set the report title for the print job. Use this method whenever you need to change the title of a report at print time.

Syntax

```
BOOL SetReportTitle ( const _TCHAR *title );
```

Parameter

title	Specifies a pointer to the new title for the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetSectionFormat

Use CRPEJob::SetSectionFormat to set the section format settings for selected sections in the specified report to the values in **CRPESectionOptions** (page 142). Use this method to provide specialized formatting for printing invoices, form letters, printing on pre-printed forms, etc.

SetSectionFormat 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 **CRPESectionOptions** (page 142).

Syntax

```
BOOL SetSectionFormat (
          short sectionCode,
          const CRPESectionOptions *options );
```

Parameters

sectionCode	Specifies the code for the report section for which you want to change format information. Use one of the PEP_XXX **Section Codes Constants** (page 174).
options	Specifies a pointer to **CRPESectionOptions** (page 142).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetSectionFormatFormula

Use CRPEJob::SetSectionFormatFormula to change the specified section format formula to the formula string that you supply as a parameter. This method will change only the text of a formula that already exists in the report; you cannot use it to add a formula.

Syntax

```
BOOL SetSectionFormatFormula (
          short sectionCode,
          short formulaName,
          CString formulaText );
```

Parameters

sectionCode	Specifies the code for the report section(s) for which you want to set formatting options. See *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.
formulaName	Specifies the name of the formula for which you want to supply a new string. Use one of the PEP_FFN_XXX **Area/Section Format Formula Constants** (page 157). See Remarks below.
formulaText	The Cstring object that contains the specified formula text when CRPEJob::SetFormula has completed successfully. See Remarks below.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- Returns PEPRR_BADFORMULANAME, if the formula name passed in does not exist, or it does not apply to the section. See **Error Codes** (page 159) for further information.
- Returns PEPERR_BADFORMULATEXT if there is an error in the formula. See **Error Codes** (page 159) for further information.
- 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.
- This method should be called before *CRPEJob::Start, Page 103*, or the results may be inconsistent or unexpected.
- Not all formula names can be applied to all sections.

CRPEJob::SetSelectionFormula

Use CRPEJob::SetSelectionFormula to set the formula text for the record selection formula. CRPEJob::SetSelectionFormula is often used with *CRPEJob::GetSelectionFormula, Page 65*, to identify and then change an existing selection formula in response to a user selection at print time.

```
BOOL SetSelectionFormula ( const _TCHAR *formulaText );
```

Parameter

formulaText	Specifies a pointer to the new selection formula to be assigned to the report.

Returns

TRUE if the call is successful.

- FALSE if the call fails or if there is an error in the formula.

CRPEJob::SetSectionHeight

Use CRPEJob::SetSectionHeight to set the section height information for the specified section. Use this method whenever you want to specify the section height for printing on pre-printed forms or printing on any other kind of document with a fixed format.

Syntax

```
BOOL SetSectionHeight (
           short sectionCode,
           short height );
```

Parameters

sectionCode	Specifies the code for the report sections for which you want to set section height information.
height	Specifies the section height, in twips.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetSQLExpression

Use CRPEJob::SetSQLExpression to enter an SQL expression for the associated print job.

Syntax

```
BOOL SetSQLExpression (
          const _TCHAR *expressionName,
          const _TCHAR *expressionText );
```

Parameters

expressionName	Specifies a pointer to the expression name.
expressionText	Specifies a pointer to the SQL expression text.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetSQLQuery

Use CRPEJob::SetSQLQuery to set the SQL query that will be sent to the database server. This method can be used with *CRPEJob::GetSQLQuery, Page 66*, to retrieve and then change the SQL query for the report.

Syntax

```
BOOL SetSQLQuery ( const _TCHAR *query );
```

Parameter

query	Specifies a pointer to the new SQL query to be sent to the server.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetTrackCursorInfo

Use CRPEJob::SetTrackCursorInfo to set information for tracking the position of the mouse cursor over the preview window. For example, this method can be used to give the user visual feedback, especially when tracking events. CRPEJob::SetTrackCursorInfo is valid only if the report was sent to a preview window and if events for the preview window have been enabled. See *CRPEJob::EnableEvent, Page 26*, for more details.

Syntax

```
BOOL SetTrackCursorInfo ( CRPETrackCursorInfo *cursorInfo );
```

Parameter

cursorInfo	Specifies a pointer to **CRPETrackCursorInfo** (page 150).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

- By default, all area track cursors are arrow cursors.
- If the preview window contains a drill-down field (database field, summary, group name, group chart), the group area will use a magnify cursor.
- Group charts in other areas will also use the magnify cursor.

CRPEJob::SetWindowOptions

Use CRPEJob::SetWindowOptions to set the options associated with a specified window object. This information is retrieved from the structure **CRPEWindowOptions** (page 153), by using *CRPEJob::GetWindowOptions, Page 67*. Use this method to customize the features of a specified window object.

Syntax

```
BOOL SetWindowOptions ( const CRPEWindowOptions *windowOptions );
```

Parameter

windowOptions	Specifies a pointer to **CRPEJob::GetWindowOptions** (page 67), which will contain the information that you want to set.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::Show...Page

Use CRPEJob::Show...Page to display the specified page of a report in the preview window. Use this method to customize how a user moves through pages in a report.

Syntax

```
BOOL ShowFirstPage ();
BOOL ShowLastPage ();
BOOL ShowNextPage ();
BOOL ShowNthPage ( short pageN );
BOOL ShowPreviousPage ();
```

Parameter

The following parameter applies only to ShowNthPage.

pageN	Specifies the page number of the report that should be displayed in the preview window.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ShowPrintControls

Use CRPEJob::ShowPrintControls to specify whether print control buttons are displayed in the preview window. Use this method to display default controls, or to hide default controls and provide customized controls.

Syntax

```
BOOL ShowPrintControls ( BOOL showControls );
```

Parameter

showControls	Set to TRUE to indicate that default Print controls are to be displayed in the preview window. Set to FALSE to indicate that no Print controls are to be displayed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::Start

Use CRPEJob::Start 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.

Syntax

```
BOOL Start ();
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SVA2T

Use helper method CRPEJob::SVA2T to convert a ValueInfo structure to UNICODE.

Syntax

```
void SVA2T (
          CRPEValueInfo *valueOut,
          PEValueInfo *valueIn );
```

Parameters

Parameter	*Description*
valueOut	Specifies a pointer to the returned **CRPEValueInfo** (page 152) structure, converted to UNICODE.
valueIn	Specifies a pointer to the **PEValueInfo** (*Seagate Crystal Reports Technical Reference Guide*) input structure.

CRPEJob::SVT2A

Use helper method CRPEJob::SVT2A to convert a ValueInfo structure to ASCII.

Syntax

```
void SVT2A (
          PEValueInfo *valueOut,
          CRPEValueInfo* valueIn );
```

Parameters

Parameter	*Description*
valueOut	Specifies a pointer to the returned **PEValueInfo** (*Seagate Crystal Reports Technical Reference Guide*) structure, converted to ASCII.
valueIn	Specifies a pointer to the **CRPEValueInfo** (page 152) input structure.

CRPEJob::TestNthTableConnectivity

Use CRPEJob::TestNthTableConnectivity to determine whether 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 confirm that everything is in order for logging on to the database table.

Syntax

```
BOOL TestNthTableConnectivity ( short tableN );
```

Parameter

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.

Returns

- TRUE if the database session, logon, and location information is all correct.
- FALSE if the connection fails.

CRPEJob::VerifyDatabase

Use CRPEJob::VerifyDatabase to verify that the database connection(s) specified in a report is (are) valid.

Syntax

```
BOOL VerifyDatabase ();
```

Returns

- TRUE if connection is valid.
- FALSE if connection is not valid or if call is not successful.

CRPEJob::ZoomPreviewWindow

Use CRPEJob::ZoomPreviewWindow to change the preview window magnification level to the specified level. Use this method when the report has been printed to a preview window and you want to set the magnification of the preview window to a specific level.

Syntax

```
BOOL ZoomPreviewWindow ( short level );
```

Parameter

level	Specifies the magnification level to which the preview window is to be zoomed. Use one of the PEP_ZOOM_XXX **Zoom Constants** (page 176).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEngine and class CRPEJob Structures

This section describes the structures associated with class CRPEngine and class CRPEJob.

CRPECloseButtonClickedEventInfo

This structure contains information about a close button-clicked event. When the close button in a preview window is clicked, a callback function will be called with EventId = PEP_CLOSE_BUTTON_CLICKED_EVENT. The fieldValueList will be released after the callback function. Make a copy of the field value list if you want to keep the field value information.

Data Members

The default for each data member is 0.

m_viewIndex	WORD	Specifies which view is going to be closed. Start from 0.
m_windowHandle	long	Specifies on which frame window handle the button is positioned.

Constructor CRPECloseButtonClickedEventInfo::CRPECloseButtonClickedEventInfo

This constructs a CRPECloseButtonClickedEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPECloseButtonClickedEventInfo ();
```

CRPEDrillOnDetailEventInfo

This structure provides PEP_DRILL_ON_DETAIL_EVENT event information.

Data Members

Member	*Type*	*Default*	*Description*
m_selectedFieldIndex	short	0	A 0-based index indicating which drill-down field was selected. Contains -1 if no field is selected.
m_windowHandle	long	0	Frame window handle where the drill on detail event happens.
m_fieldValueList	structure	NULL	Specifies a pointer to an array field value of the CRPEFieldValueInfo structure's field value list.
m_nFieldValue	short	0	Number of field value in field value of list. For example, 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 a database field, group name field, summary field, or a formula field. If the user clicks a text object, graph, picture, ole, subreport, special var field, or database memo or blob field, the selectedFieldIndex will return -1.

Constructor
CRPEDrillOnDetailEventInfo::CRPEDrillOnDetailEventInfo

This constructs a CRPEDrillOnDetailEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEDrillOnDetailEventInfo ();
```

CRPEDrillOnGroupEventInfo

This structure specifies whether to drill on group information when PEP_DRILL_ON_GROUP_EVENT happens.

Data Members

Member	*Type*	*Default*	*Description*
m_drillType	WORD	0	Specifies the type of drill down that is used. Use one of the following PEP_DE_XXX constants.
	Constant		***Description***
	PEP_DE_ON_GROUP		Drill down on a group summary or subtotal.
	PEP_DE_ON_GROUPTREE		Drill down a Group Tree node.
	PEP_DE_ON_GRAPH		Drill down a group chart object.
	PEP_DE_ON_MAP		Drill down on a map region.
	PEP_DE_ON_SUBREPORT		Drill down on a subreport.
m_windowHandle	long	0	Pointer to an array of group names. The list specifies the group path of the selected group. The memory is freed after the callback function is called.
m_groupList	TCHAR	NULL	Specifies a pointer to an array of group names in the report when drilling on a group summary, a Group Tree, a chart, or a map. Specifies a pointer to a single element array containing the subreport name when drilling on a subreport. This memory is freed after the callback function is called.
m_groupLevel	WORD	0	The number of the group name in the group list.

Remarks

The groupList will be freed after the callback function. If you want to keep the groupList, make a copy of it.

Constructor CRPEDrillOnGroupEventInfo::CRPEDrillOnGroupEventInfo

This constructs a CRPEDrillOnGroupEventInfo structure object. It assigns default values to the members of the structure.

Syntax

```
CRPEDrillOnGrupEventInfo ();
```

CRPEEnableEventInfo

This structure specifies which group event is enabled or disabled. All events are disabled by default.

Data Members

- Each data member is set by the corresponding constructor parameter.
- Each data member can be set to TRUE, FALSE, or PE_UNCHANGED.
- The default for each data member is FALSE.

m_startStopEvent	short	Specifies a StartStopEvent.
m_readingRecordEvent	short	Specifies a ReadingRecordEvent.
m_printWindowButtonEvent	short	Specifies a PrintWindowButtonEvent.
m_drillEvent	short	Specifies a DrillEvent.
m_closePrintWindowEvent	short	Specifies a ClosePrintWindowEvent.
m_activatePrintWindowEvent	short	Specifies an ActivatePrintWindowEvent.
m_fieldMappingEvent	short	Specifies a FieldMappingEvent.
m_mouseClickEvent	short	Specifies a MouseClickEvent.
m_hyperlinkEvent	short	Specifies a Hyperlink Event.
launchSeagateAnalysisEvent	short	Specifies a launch Seagate Analysis Event.

Remarks

All events are disabled by default. Use *CRPEJob::EnableEvent, Page 26*, to enable the desired event.

Constructor CRPEEnableEventInfo::CRPEEnableEventInfo

This constructs a CRPEEnableEventInfo 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.

Constructor Syntax (Default)

```
CRPEEnableEventInfo ();
```

Constructor Syntax

```
CRPEEnableEventInfo (
          short startStopEvent,
          short readingRecordEvent,
          short printWindowButtonEvent,
          short drillEvent,
          short closePrintWindowEvent,
          short activatePrintWindowEvent,
          short fieldMappingEvent,
          short mouseClickEvent,
          short hyperlinkEvent,
          short launchSeagateAnalysisEvent );
```

CRPEExportOptions

This structure is used to retrieve and set the export options of a print job. It is used by member function *CRPEJob::ExportTo, Page 28*.

Data Members

Each data member is set by the corresponding constructor parameter.

Member	*Type*	*Default*	*Description*
m_formatDLLName	_TCHAR	'\0'	Specifies the name of the format DLL (of length PEP_DLL_NAME_LEN = 64) that contains the export format to be used. Select a DLL from the **Export Format DLL** (page 112) table below.
m_formatType	DWORD	0	Specifies the export format to be used. Select a DLL from the **Export Format Type** (page 112) table below.

Member	*Type*	*Default*	*Description*
m_formatOptions	void	NULL	Specifies additional export information specific to the format type being used. This parameter is required for certain format types only. If you assign the default NULL to formatOptions, the Seagate Crystal Report Engine will automatically prompt the user for format information if it is needed. Alternatively, you can choose one from the **Export Format Options** (page 113) table below.
m_destinationDLLName	_TCHAR	'\0'	Specifies the name of the destination DLL (of length PEP_DLL_NAME_LEN = 64) that contains the destination type to be used. Select a destination DLL from the **Destination DLL Name** (page 113) table below.
m_destinationType	DWORD	0	Specifies the destination type of the exported report. Select a destination type from the **Destination Type** (page 113) table below.
m_destinationOptions	void	NULL	Provides additional information specific to the export destination type. If you assign the default NULL to destinationOptions, the Seagate Crystal Report Engine will automatically prompt the user for destination information if it is needed. Alternatively, you can choose one from the **Export Destination Options Type** (page 113) table below.
m_nFormatOptionsBytes	WORD	0	Automatically assigned by CRPEJob::GetExportOptions, Page 31. Unused by CRPEJob::ExportTo, Page 28.
m_nDestinationOptions Bytes	WORD	0	Automatically assigned by CRPEJob::GetExportOptions, Page 31. Unused by CRPEJob::ExportTo, Page 28.

CRPEExportOptions Options Tables

Export Format DLL

Export Format	*DLL*
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 (WK3)	uxfwks.dll
Excel 4.0	uxfxls.dll

Export Format Type

Export Format	*Format Type*
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

Export Format	*Format Type*
Tab Separated Text Format	UXFTabbedTextType
Lotus 1-2-3 (WK3)	UXFLotusWk3Type
Excel 4.0	UXFXls4Type

Export Format Options

Export Format	*Format Type*
Data Interchange Format	UXFDIFOptions
Record Style Format (columns)	UXFRecordStyleOptions
Comma Separated Values (CSV)	UXFCommaTabSeparatedOptions
Tab Separated Values	UXFCommaTabSeparatedOptions
Character Separated Values	UXFCharSeparatedOptions

Destination DLL Name

Report Destination	*DLL*
Disk File	uxddisk.dll
E-mail (MAPI)	uxdmapi.dll
E-mail (VIM)	uxdvim.dll

Destination Type

Report Destination	*Destination Type*
Disk File	UXDDiskType
E-mail (MAPI)	UXDMapiType
E-mail (VIM)	UXDVIMType

Export Destination Options Type

Report Destination	*Structure*
Disk File	UXDDiskOptions
E-mail (MAPI)	UXDMAPIOptions
E-mail (VIM)	UXDVIMOptions

Constructor CRPEExportOptions::CRPEExportOptions

This 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.

Constructor Syntax (Default)

```
struct CRPEExportOptions ();
```

Constructor Syntax

```
struct CRPEExportOptions (
          const _TCHAR *formatDLLName,
          DWORD formatType,
          void *formatOptions,
          const _TCHAR *destinationDLLName,
          DWORD destinationType,
          void *destinationOptions );
```

CRPEFieldMappingEventInfo

This structure provides field mapping event information.

Data Members

Each data member is set by the corresponding constructor parameter.

Member	*Type*	*Default*	*Description*
m_reportFields	WORD	NULL	An array of fields in the report. See Remarks below.
m_nReportFields	WORD	0	The size of the reportFields array.
m_databaseFields	WORD	NULL	An array of fields in the new database. See Remarks below.
m_nDatabaseFields	WORD	0	The size of the databaseFields array.

Remarks

- Member reportFields is an array of fields in the report of size nReportFields. The user must modify the 'mappingTo' of each newly mapped field by assigning the value of the index of a field in the array databaseFields.
- Member databaseFields is an array of fields in the new database file of size nDatabaseFields.
- CRPEReportFieldMappingInfo.mappingTo must be modified for each newly mapped field by assigning the value of the index of a field in the databaseFields array.

Constructor CRPEFieldMappingEventInfo::CRPEFieldMappingEventInfo

This constructs a CRPEFieldMappingEventInfo 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.

Constructor Syntax (Default)

```
CRPEFieldMappingEventInfo ();
```

Constructor Syntax

```
CRPEFieldMappingEventInfo (
          CRPEReportFieldMappingInfo *reportFields,
          WORD nReportFields,
          CRPEReportFieldMappingInfo *databaseFields,
          WORD nDatabaseFields );
```

CRPEFieldValueInfo

This structure specifies a field value in the fieldValueList of the structure **CRPEDrillOnDetailEventInfo** (page 107).

Data Members

Member	*Type*	*Default*	*Description*
m_ignored	WORD	0	For 4 byte alignment. Ignore.
m_fieldName	_TCHAR	'\0'	Specifies the field name (of length PEP_FIELD_NAME_LEN = 512).
m_fieldValue	CRPEValueInfo	CRPEValueInfo()	CRPEValueInfo's field value for the selected Details area specified by the fieldName member variable.

Remarks

A selected Details area corresponds to a database record.

Constructor CRPEFieldValueInfo::CRPEFieldValueInfo

This constructs a CRPEFieldValueInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEFieldValueInfo ();
```

CRPEFontColorInfo

CRPEFontColorInfo contains information regarding the fonts used in a chart.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default value for each data member is 0, except for m_faceName and m_color.

Member	*Type*	*Description*
m_faceName	_TCHAR	Specifies the font face name (of length PEP_FACE_NAME_LEN = 64). Use an empty string ("") for no change. Default is '\0'.
m_fontFamily	short	Specifies the font family. Use FF_DONTCARE for no change.
m_fontPitch	short	Specifies the font pitch. Use DEFAULT_PITCH for no change.
m_charSet	short	Specifies the font charset. Use DEFAULT_CHARSET for no change.
m_pointSize	short	Specifies the desired point size for the selected font. Use this member or member twipSize to specify the font size. If both members are non-zero, then this member will be ignored and twipSize will be used. Pass 0 for both twipSize and pointSize for no change.
m_isItalic	short	Boolean. If TRUE, the text is italic. Use PEP_UNCHANGED for no change.
m_isUnderlined	short	Boolean. If TRUE, the text is underlined. Use PEP_UNCHANGED for no change.
m_isStruckOut	short	Boolean. If TRUE, the text is struckout. Use PEP_UNCHANGED for no change.
m_weight	short	Specifies the font weight. Use 0 for no change.
m_color	COLORREF	Specifies Windows COLORREF or PEP_UNCHANGED_COLOR or no change. Default is PEP_UNCHANGED_COLOR.
m_twipSize	short	Specifies the font size, in twips. Use this member or member pointSize to specify the font size. If both members are non-zero, then this member will be used and pointSize will be ignored. Pass 0 for both twipSize and pointSize for no change.

Constructor CRPEFontColorInfo::CRPEFontColorInfo

This constructs a CRPEFontColorInfo 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.

Constructor Syntax (Default)

```
CRPEFontColorInfo ()
```

Constructor Syntax

```
CRPEFontColorInfo (
          const _TCHAR *faceName,
          short fontFamily,
          short fontPitch,
          short charSet,
          short pointSize,
          short isItalic,
          short isUnderlined,
          short isStruckOut,
          short weight,
          COLORREF color
          short twipSize );
```

CRPEFormulaSyntax

CRPEFormulaSyntax contains information related to the syntax of the specified formula.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default for each member of the m_formula syntax array is 0 (PEP_FST_CRYSTAL).

Member	*Type*	*Description*
m_formulaSyntax	short	Specifies an array (of size PEP_FS_SIZE = 2) which contains the formula syntax information as PEP_FST_XXX **Formula Syntax Constants** (page 164) constants or PEP_UNCHANGED for no change.

Constructor CRPEFormulaSyntax::CRPEFormulaSyntax

This constructs a CRPEFormulaSyntax structure object. It assigns default values to the members of the structure.

Constructor Syntax (Default)

```
CRPEFormulaSyntax ()
```

CRPEGeneralPrintWindowEventInfo

This structure specifies general preview window event information and is used for the following events.

- PRINT_WINDOW_EVENT
- PEP_CLOSE_PEP_PRINT_BUTTON_CLICKED_EVENT
- PEP_EXPORT_BUTTON_CLICKED_EVENT
- PEP_FIRST_PAGE_BUTTON_CLICKED_EVENT
- PEP_PREVIOUS_PAGE_BUTTON_CLICKED_EVENT
- PEP_NEXT_PAGE_BUTTON_CLICKED_EVENT
- PEP_LAST_PAGE_BUTTON_CLICKED_EVENT
- PEP_CANCEL_BUTTON_CLICKED_EVENT
- PEP_PRINT_SETUP_BUTTON_CLICKED_EVENT
- PEP_REFRESH_BUTTON_CLICKED_EVENT
- PEP_ACTIVATE_PRINT_WINDOW_EVENT
- PEP_DEACTIVATE_PRINT_WINDOW_EVENT

Data Members

The default for each data member is 0.

m_ignored	WORD	For 4 byte alignment. Ignore.
m_windowHandle	long	Frame window handle where the event happens.

Constructor CRPEGeneralPrintWindowEventInfo::CRPEGeneralPrintWindowEventInfo

This constructs a CRPEGeneralPrintWindowEventInfo structure object.

Constructor Syntax (Default)

```
CRPEGeneralPrintWindowEventInfo ();
```

CRPEGraphAxisInfo

The CRPEGraphAxisInfo structure contains information about how the axis of a chart are calculated and displayed.

Data Members

Each data member is set by the corresponding constructor parameter.

The default value for each data member is 0.

Member	*Type*	*Description*
m_groupAxisGridLine	short	Specifies how gridlines are displayed on the chart. Use one of the PEP_GGT_XXX **Graph Gridline Constants** (page 165), or PEP_UNCHANGED for no change.
m_dataAxisYGridLine	short	Specifies how gridlines are displayed on the chart. Use one of the PEP_GGT_XXX **Graph Gridline Constants** (page 165), or PEP_UNCHANGED for no change.
m_dataAxisY2GridLine	short	Specifies how gridlines are displayed on the chart. Use one of the PEP_GGT_XXX **Graph Gridline Constants** (page 165), or PEP_UNCHANGED for no change.
m_seriesAxisGridline	short	Specifies how gridlines are displayed on the chart. Use one of the PEP_GGT_XXX **Graph Gridline Constants** (page 165), or PEP_UNCHANGED for no change.
m_dataAxisYMinValue	double	Specifies the minimum value for the axis.
m_dataAxisYMaxValue	double	Specifies the maximum value for the axis.
m_dataAxisY2MinValue	double	Specifies the minimum value for the axis.
m_dataAxisY2MaxValue	double	Specifies the maximum value for the axis.
m_seriesAxisMinValue	double	Specifies the minimum value for the axis.
m_seriesAxisMaxValue	double	Specifies the maximum value for the axis.
m_dataAxisYNumber Format	short	Specifies the format for the display of numeric values on the chart. Use one of the PEP_GNF_XXX **Chart Number Format Constants** (page 167), or PEP_UNCHANGED for no change.
m_dataAxisY2Number Format	short	Specifies the format for the display of numeric values on the chart. Use one of the PEP_GNF_XXX **Chart Number Format Constants** (page 167), or PEP_UNCHANGED for no change.
m_seriesAxisNumber Format	short	Specifies the format for the display of numeric values on the chart. Use one of the PEP_GNF_XXX **Chart Number Format Constants** (page 167), or PEP_UNCHANGED for no change.
m_dataAxisYAutoRange	short	Boolean, or PEP_UNCHANGED for no change. If TRUE, the axis will autorange.
m_dataAxisY2Auto Range	short	Boolean, or PEP_UNCHANGED for no change. If TRUE, the axis will autorange.
m_seriesAxisAutoRange	short	Boolean, or PEP_UNCHANGED for no change. If TRUE, the axis will autorange.
m_dataAxisYAutomatic Division	short	Use one of the PEP_ADM_XXX **Graph Axis Division Method Constants** (page 165), or PEP_UNCHANGED for no change.
m_dataAxisY2Automatic Division	short	Use one of the PEP_ADM_XXX **Graph Axis Division Method Constants** (page 165), or PEP_UNCHANGED for no change.

Member	*Type*	*Description*
m_seriesAxisAutomatic Division	short	Use one of the PEP_ADM_XXX **Graph Axis Division Method Constants** (page 165), or PEP_UNCHANGED for no change.
m_dataAxisYManual Division	long	Valid only if the corresponding axis m_dataAxisYAutomaticDivision is FALSE.
m_dataAxisY2Manual Division	long	Valid only if the corresponding axis m_dataAxisY2AutomaticDivision is FALSE.
m_seriesAxisManual Division	long	Valid only if the corresponding axis m_seriesAxisAutomaticDivision is FALSE.
m_dataAxisYAutoScale	short	Boolean, or PEP_UNCHANGED for no change. Specifies whether the axis should autoscale.
m_dataAxisY2AutoScale	short	Boolean, or PEP_UNCHANGED for no change. Specifies whether the axis should autoscale.
m_seriesAxisAutoScale	short	Boolean, or PEP_UNCHANGED for no change. Specifies whether the axis should autoscale.

Constructor CRPEGraphAxisInfo::CRPEGraphAxisInfo

This constructs a CRPEGraphAxisInfo 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.

Constructor Syntax (Default)

```
CRPEGraphAxisInfo ()
```

Constructor Syntax

```
CRPEGraphAxisInfo (
          WORD StructSize,
          short groupAxisGridLine,
          short dataAxisYGridLine,
          short dataAxisY2GridLine,
          short seriesAxisGridline,
          double dataAxisYMinValue,
          double dataAxisYMaxValue,
          double dataAxisY2MinValue,
          double dataAxisY2MaxValue,
          double seriesAxisMinValue,
          double seriesAxisMaxValue,
          short dataAxisYNumberFormat,
          short dataAxisY2NumberFormat,
          short seriesAxisNumberFormat,
          short dataAxisYAutoRange,
```

```
short dataAxisY2AutoRange,
short seriesAxisAutoRange,
short dataAxisYAutomaticDivision,
short dataAxisY2AutomaticDivision,
short seriesAxisAutomaticDivision,
long dataAxisYManualDivision,
long dataAxisY2ManualDivision,
long seriesAxisManualDivision,
short dataAxisYAutoScale,
short dataAxisY2AutoScale,
short seriesAxisAutoScale );
```

CRPEGraphOptionInfo

The CRPEGraphOptionInfo structure contains information on several options available with charts.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default for each data member is 0.
- For each data member, use PEP_UNCHANGED for no change.

Member	*Type*	*Description*
m_graphColour	short	Specifies the chart color. Use one of thePEP_GCR_XXX **Graph Color Constants** (page 167).
m_showLegend	short	Boolean. If TRUE, the chart legend will be visible. If FALSE, the chart legend will be hidden.
m_legendPosition	short	Specifies the position of the legend on the chart if m_showLegend is TRUE. Use one of the PEP_GLP_XXX **Graph Placement Constants** (page 165).
m_pieSize	short	Specifies the size of the slice on a pie or a doughnut chart. Use one of the PEP_GPS_XXX **Graph Pie Size Constants** (page 166).
m_detachedPieSlice	short	Specifies the detachment of the slice on a pie or doughnut chart. Use one of the PEP_GDPS_XXX **Graph Detached Pie Slice Constants** (page 166).
m_barSize	short	Specifies the bar size on bar charts. Use one of the PEP_GBS_XXX **Graph Bar Size Constants** (page 165).
m_verticalBars	short	Boolean. If TRUE, there will be vertical bars on bar charts.
m_markerSize	short	Specifies the size of the marker on line and bar charts. Use one of the PEP_GMS_XXX **Graph Marker Size Constants** (page 166).
m_markerShape	short	Specifies the shape of the marker on line and bar charts. USe one of the PEP_GMSP_XXX **Graph Marker Shape Constants** (page 166).

Member	*Type*	*Description*
m_dataPoints	short	Specifies how data points will be shown. Use one of the PEP_GDP_XXX **Chart Data Point Constants** (page 167).
m_dataValueNumber Format	short	Specifies the number format for values on the chart. Use one of the PEP_GNF_XXX **Chart Number Format Constants** (page 167).
m_viewingAngle	short	Specifies the viewing angle with which a 3D chart will be shown. Use one of the PEP_GVA_XXX **Graph Viewing Angle Constants** (page 168).
m_legendLayout	short	Specifies the chart legend layout. Use one of the PEP_GLL_XXX **Graph Legend Layout Constants** (page 166). Default = PEP_GLL_PERCENTAGE.

Constructor CRPEGraphOptionInfo::CRPEGraphOptionInfo

This constructs a CRPEGraphOptionInfo 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.

Constructor Syntax (Default)

```
CRPEGraphOptionInfo ();
```

Constructor Syntax

```
CRPEGraphOptionInfo (
          short graphColour,
          short showLegend,
          short legendPosition,
          short pieSize,
          short detachedPieSlice,
          short barSize,
          short verticalBars,
          short markerSize,
          short markerShape,
          short dataPoints,
          short dataValueNumberFormat,
          short viewingAngle,
          short legendLayout )
```

CRPEGraphTypeInfo

The CRPEGraphTypeInfo structure contains information on what chart type is specified.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default for each member is 0.

m_graphType	short	Specifies the chart type. Uses one of the PEP_GT_XXX **Graph Type and Subtype Constants** (page 169), or PEP_UNCHANGED for no change.
m_graphSubtype	short	Specifies the chart subtype. Uses one of the PEP_GST_XXX **Graph Type and Subtype Constants** (page 169), or PEP_UNCHANGED for no change.

Constructor CRPEGraphTypeInfo::CRPEGraphTypeInfo

This constructs a CRPEGraphTypeInfo 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.

Constructor Syntax (Default)

```
CRPEGraphTypeInfo ();
```

Constructor Syntax

```
CRPEGraphTypeInfo (
          short graphType,
          short graphSubtype );
```

CRPEGroupOptions

This structure contains information about several options available with report groups. This information is used by *CRPEJob::GetGroupOptions, Page 40*, to retrieve current options and by *CRPEJob::SetGroupOptions, Page 86*, to pass new option settings.

Data Members

Each data member is set by the corresponding constructor parameter.

Member	*Type*	*Default*	*Description*
m_condition	short	0	Specifies the condition setting for the selected group section. See Remarks below.
m_fieldName	_TCHAR	'\0'	Specifies the field name of the group field (of size PEP_FIELD_NAME_LEN = 512), or remains empty for no change.

Member	*Type*	*Default*	*Description*
m_sortDirection	short	0 Descending	Specifies the sort direction. Uses one of the PEP_SF_XXX **Sort Order Constants** (page 174) or PEP_UNCHANGED for no change. For group conditions only, use PEP_SF_ORIGINAL for original order or PEP_SF_SPECIFIED for read-only specified sort order.
m_repeatGroup Header	short	0	BOOL value, or PEP_UNCHANGED for no change.
m_keepGroup Together	short	0	BOOL value, or PEP_UNCHANGED for no change.
m_topOrBottom NGroups	short	0	Use one of the following PEP_GO_TBN_XXX constants or PEP_UNCHANGED for no change.
	Constant		*Description*
	PEP_GO_TBN_ALL_ GROUPS_UNSORTED		There is no group sorting or Top/Bottom N for this level of grouping.
	PEP_GO_TBN_ALL_ GROUPS_SORTED		There is group sorting, but not Top/Bottom N.
	PEP_GO_TBN_TOP_ N_GROUPS		Top N groups will be selected.
	PEP_GO_TBN_ BOTTOM_N_GROUPS		Bottom N groups will be selected.
Member	*Type*	*Default*	*Description*
m_topOrBottom NSortFieldName	_TCHAR	'\0'	Specifies the name of the summary field (of size PEP_FIELD_NAME_LEN = 512) by which the groups are ordered, or remains empty for no change.
m_nTopOr BottomGroups	short	0 Keep all groups	Number of groups to select.
m_discardOther Groups	short	0	Determines whether the remaining groups are collected into an Others group or discarded. BOOL value, or PEP_UNCHANGED for no change.
hierarchicalSorting	short	FALSE	Boolean. Specifies whether or not to do hierarchial sorting.
*instanceIDField	_TCHAR	'\0'	Specifies a pointer to the name of the instance ID field (of size PEP_FIELD_NAME_LEN = 512) for hierarchial sorting.
*parentIDField	_TCHAR	'\0'	Specifies a pointer to the name of the parent ID field (of size PEP_FIELD_NAME_LEN = 512) for hierarchial sorting.

Member	*Type*	*Default*	*Description*
groupIndent	long	0	Specifies the indent for the group, in twips.

Remarks

- For data member m_condition, use PEP_GC_TYPEMASK and PEP_GC_CONDITIONMASK for decoding the group condition and a PEP_GC_XXX constant or PEP_UNCHANGED for setting the group condition.
- If topOrBottomNGroups is PEP_GO_TBN_TOP_N_GROUPS or PEP_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 it is deleted) if topOrBottomNSortFieldName is empty.

Constructor CRPEGroupOptions::CRPEGroupOptions

This constructs a CRPEGroupOptions 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.

Constructor Syntax (Default)

```
CRPEGroupOptions ();
```

Constructor Syntax

```
CRPEGroupOptions (
          short condition,
          _TCHAR *fieldName,
          short  sortDirection,
          short  repeatGroupHeader,
          short  keepGroupTogether,
          short  topOrBottomNGroups,
          _TCHAR *topOrBottomNSortFieldName,
          short  nTopOrBottomGroups,
          short  discardOtherGroups,
          short  hierarchicalSorting
          _TCHAR instanceIDField
          _TCHAR parentIDField
          long   groupIndent );
```

CRPEGroupTreeButtonClickedEventInfo

This structure provides information about the group tree button clicked event, when the callback function is called with the event ID equal to PEP_GROUP_TREE_BUTTON_CLICKED_EVENT.

Data Members

Member	*Type*	*Default*	*Description*
m_visible	short	FALSE	Indicates whether the group tree is shown or hidden.
m_windowHandle	long	0	Frame window handle where the event happens.

Constructor CRPEGroupTreeButtonEventInfo:CRPEGroupTreeButtonClickedEventInfo

This constructs a CRPEGroupTreeButtonClickedEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEGroupTreeButtonClickedEventInfo ();
```

CRPEHyperlinkEventInfo

This structure provides information about a selected hyperlink.

Data Members

Member	*Type*	*Default*	*Description*
m_ignored	WORD	0	For 4 byte alignment. Ignore.
m_windowHandle	long	0	HWND specifies the handle of the window where the event happened.
m_hyperlinkText	_TCHAR	'\0'	Specifies a pointer to the hyperlink text (of length PEP_HYPERLINK_LEN= 256) associated with the object. The memory pointed to by hyperlinkText is freed after calling the callback function.

Constructor CRPEHyperlinkEventInfo::CRPEHyperlinkEventInfo

This constructs a CRPEHyperlinkEventInfo 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.

Constructor Syntax (Default)

```
CRPEHyperlinkEventInfo ();
```

Constructor Syntax

```
CRPEHyperlinkEventInfo (
          long windowHandle,
          _TCHAR *hyperlinkText );
```

CRPEJobInfo

This structure is used by *CRPEJob::GetJobStatus, Page 41*. Status information concerning the job is returned through this structure.

Data Members

Member	*Type*	*Default*	*Description*
m_numRecordsRead	DWORD	0	Specifies the number of records actually processed.
m_numRecordsSelected	DWORD	0	Specifies the number of records selected for inclusion in the report out of the total number of records read.
m_numRecordsPrinted	DWORD	0	Specifies the number of records actually printed.
m_displayPageN	WORD	0	Specifies the page number of the currently displayed page in the preview window.
m_latestPageN	WORD	0	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	WORD	0	Specifies the number of the starting page.
m_printEnded	BOOL	FALSE	Specifies whether the printing process is completed.

Constructor CRPEJobInfo::CRPEJobInfo

This constructs a CRPEJobInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax (Default)

```
CRPEJobInfo ();
```

CRPELaunchSeagateAnalysisEventInfo

This structure provides information about a Launch Seagate Analysis Event.

Data Members

Member	*Type*	*Default*	*Description*
m_ignored	WORD	0	For 4 byte alignment. Ignore.
m_windowHandle	long	0	HWND specifies the handle of the window where the event happened.
m_pathFile	_TCHAR	'\0'	Specifies a pointer to the path and filename (of length PEP_PATHFILE_LEN = 256) of the temporary report. The memory pointed to by pathFile is freed after calling the callback function.

Constructor CRPELaunchSeagateAnalysisEventInfo::CRPELaunchSeagateAnalysisEventInfo

This constructs a CRPELaunchSeagateAnalysisEventInfo 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.

Constructor Syntax (Default)

```
CRPELaunchSeagateAnalysisEventInfo ();
```

Constructor Syntax

```
CRPELaunchSeagateAnalysisEventInfo (
          long windowHandle,
          _TCHAR *pathFile );
```

CRPELogOnInfo

This structure contains SQL connection information and is used by the following member functions.

- *CRPEngine::LogOnServer, Page 8*
- *CRPEngine::LogOffServer, Page 8*
- *CRPEJob::GetNthTableLogOnInfo, Page 56*
- *CRPEJob::SetNthTableLogOnInfo, Page 91*

Data Members

- Each data member is set by the corresponding constructor parameter.
- Specify an empty string ("") to use the value already set in the report.

- Specify a non-empty string (for example, "Server A") to override a value in the report.
- All strings are null-terminated.
- The default for each data member is '\0'.
- For Netware SQL, pass the dictionary path name in serverName and the data path name in databaseName.

m_serverName	_TCHAR	Specifies the logon name (of length PEP_SERVERNAME_LEN = 128) for the server used to create the report.
m_databaseName	_TCHAR	Specifies the logon name (of length PEP_DATABASENAME_LEN = 128) for the database used to create the report.
m_userID	_TCHAR	Specifies the user ID (of length PEP_USERID_LEN = 128) necessary to log on to the server.
m_password	_TCHAR	Specifies the password (of length PEP_PASSWORD_LEN = 128) necessary to log on to the server. Password is undefined when getting information from the report.

Constructor CRPELogOnInfo::CRPELogOnInfo

This 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.

Constructor Syntax (Default)

```
CRPELogOnInfo ();
```

Constructor Syntax

```
CRPELogOnInfo (
          const _TCHAR *serverName,
          const _TCHAR *databaseName,
          const _TCHAR *userID,
          const _TCHAR *password );
```

CRPEMouseClickEventInfo

The CRPEMouseClickEventInfo structure contains information related to mouse click events.

Data Members

Each data member is set by the corresponding constructor parameter.

Member	*Type*	*Default*	*Description*
m_windowHandle	long	0	windowHandle.

Member	*Type*	*Default*	*Description*
m_clickAction	UNIT	0	The click action, button down or up.
m_clickFlags	UNIT	0	Any combination of one or more of the following mouse click flag (virtual key state-masks) constants.
	Constant	***Value***	***Description***
	PEP_CF_NONE	0x0000	No button clicked.
	PEP_CF_LBUTTON	0x0001	Left mouse button clicked.
	PEP_CF_RBUTTON	0x0002	Right mouse button clicked.
	PEP_CF_SHIFTKEY	0x0004	Shift key depressed.
	PEP_CF_CONTROLKEY	0x0008	Control key depressed.
	PEP_CF_MBUTTON	0x0010	Middle mouse button clicked.
Member	***Type***	***Default***	***Description***
m_xOffset	int	0	The x-coordinate of the mouse click in pixels.
m_yOffset	int	0	The y-coordinate of the mouse click in pixels.
m_fieldValue	CRPEValueInfo	NULL	Specifies the CRPEValueInfo structure containing the value of the object at the click point. See Remarks below.
m_objectHandle	DWORD	0	Design view object handle.
m_sectionCode	short	0	The code for the section in which the click occurred.

Remarks

Member fieldValue represents a pointer to a CRPRValueInfo structure containing the value of the object at the click point if it is a field object (excluding MEMO and BLOB fields); otherwise valueType element = PEP_VI_NOVALUE.

Constructor CRPEMouseClickEventInfo::CRPEMouseClickEventInfo

This constructs a CRPEMouseClickEventInfo 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.

Constructor Syntax (Default)

```
CRPEMouseClickEventInfo ();
```

Constructor Syntax

```
CRPEMouseClickEventInfo (
          long windowHandle,
          UINT clickAction,
          UINT clickFlags,
          int xOffset,
          int yOffset,
          CRPEValueInfo *fieldValue,
          DWORD objectHandle,
          short sectionCode );
```

CRPEParameterFieldInfo

The CRPEParameterFieldInfo structure contains information related to parameter fields in a report. This structure is used by *CRPEJob::GetNthParameterField, Page 52*, to get information about a specific parameter field and by *CRPEJob::SetNthParameterField, Page 89*, to change a specific parameter field.

Data Members

- Each data member is set by the corresponding constructor parameter except as noted below.
- If you wish to set a parameter to NULL, set the CurrentValue to CRWNULL. CRWULL is of Type String and is independent of the data type of the parameter.

Member	***Type***	***Default***	***Description***
m_ValueType	WORD	0	Specifies the data type of the parameter field. The Seagate Crystal Report Engine supports the following data types and associated PEP_PF_XXX constants.
		Data Type	***Constant***
		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
Member	***Type***	***Default***	***Description***
m_DefaultValue Set	WORD	0	Specifies whether a default value is set for the parameter field. The value can be either TRUE if you want to change the default value or FALSE if you do not.

Member	*Type*	*Default*	*Description*
m_CurrentValue Set	WORD	0	Indicates whether a new value is being assigned to the parameter field. Use TRUE to indicate that a new value is set, FALSE to indicate no change.
m_Name	_TCHAR	'\0'	Specifies the name of the parameter field (of length PEP_PF_NAME_LEN = 256, NULL-terminated) to be assigned a new value.
m_Prompt	_TCHAR	'\0'	Specifies the prompting text (of length PEP_PF_PROMPT_LEN = 256, NULL-terminated), if any, that will appear when the user runs the report for the first time or refreshes the data.
m_DefaultValue	_TCHAR	'\0'	Specifies the default value assigned to the parameter field. If the DefaultValueSet member is FALSE, this value is meaningless. DefaultValue can be a Number, Date, DateTime, Time, Boolean, or String (of length PEP_PF_VALUE_LEN = 256).
m_CurrentValue	_TCHAR	'\0'	Specifies the current value assigned to the parameter field. If CurrentValueSet is FALSE, this value is meaningless. CurrentValue can be a Number, Date, DateTime, Time, Boolean, or String (of length PEP_PF_VALUE_LEN = 256).
m_ReportName	TCHAR	'\0'	The name of the report where the field belongs (used only with *CRPEJob::GetNthParameterField, Page 52*, and NewParameterField; set only by the default constructor).
m_needsCurrent Value	WORD	0	Returns FALSE if the parameter is linked, not in use, or has current value set (set only by the default constructor).
m_isLimited	WORD	0	Specifies whether or not the parameter field is limited in length (Strings) or by a range (other data types).
m_MinSize	double	0	Depending on the value type, sets the minimum length of the string or minimum numeric value.
m_MaxSize	double	0	Depending on the value type, sets the maximum length of the string or maximum numeric value.
m_EditMask	_TCHAR	\0'	An edit mask (of length PEP_PF_EDITMASK_LEN = 256) that restricts what may be entered for string parameters.
m_isHidden	WORD	FALSE	TRUE if an essbase sub var.

Constructor CRPEParameterFieldInfo::CRPEParameterFieldInfo

This 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.

Constructor Syntax (Default)

```
CRPEParameterFieldInfo ();
```

Constructor Syntax

```
CRPEParameterFieldInfo (
          WORD ValueType,
          WORD DefaultValueSet,
          WORD CurrentValueSet,
          const _TCHAR *Name,
          const _TCHAR *Prompt,
          const _TCHAR *DefaultValue,
          const _TCHAR *CurrentValue,
          WORD isLimited,
          double MinSize,
          double MaxSize,
          const _TCHAR *EditMask,
          WORD isHidden );
```

CRPEParameterPickListOption

CRPEParameterPickListOption structure contains information related to parameter sort methods.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default value for each data member is 0.

Member	*Type*	*Description*
m_showDescOnly	short	Boolean, or PEP_UNCHANGED for no change.
m_sortMethod	short	Specifies the sort method. Use one of the PEP_OR_XXX **Sort Method Constants** (page 174), or PEP_UNCHANGED for no change.
m_sortBasedOnDesc	short	Boolean, or PEP_UNCHANGED for no change.

Constructor CRPEParameterPickListOption::CRPEParameterPickListOption

This constructs a CRPEParameterPickListOption 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.

Constructor Syntax (Default)

```
CRPEParameterPickListOption ()
```

Constructor Syntax

```
CRPEParameterPickListOption (
          WORD  StructSize,
          short showDescOnly,
          short sortMethod,
          short sortBasedOnDesc );
```

CRPEParameterValueInfo

The CRPEParameterValueInfo structure contains information related to parameter values in a report. This structure is used with the methods *CRPEJob::GetParameterValueInfo, Page 59*, and *CRPEJob::SetParameterValueInfo, Page 95*, to set specific parameter value information.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default value for each data member is PEP_UNCHANGED.

m_isNullable	short	Specifies whether or not a field is nullable in a report. Set this variable to TRUE if you want a nullable report field, FALSE if you do not, or PEP_UNCHANGED to keep the existing setting.
m_disallowEditing	short	Specifies whether or not a report field can be edited. Set this variable to TRUE if you want to disallow report field editing, FALSE if you do not, or PEP_UNCHANGED to keep the existing setting.
m_allowMultipleValues	short	Specifies whether or not a field can have multiple values in a report. Set this variable to TRUE if you want to allow multiple values in a report field, FALSE if you do not, or PEP_UNCHANGED to keep the existing setting.
m_hasDiscreteValues	short	Specifies whether or not a field has discrete values in a report. Uses one of the following PEP_DR_XXX constants, or PEP_UNCHANGED to keep existing setting.
		Constant
		PEP_DR_HASRANGE
		PEP_DR_HASDISCRETE
		PEP_DR_HASDISCRETEANDRANGE
m_partOfGroup	short	Specifies whether or not the parameter field is a member of a group. Set this variable to TRUE if a report field is part of a group, FALSE if not, or PEP_UNCHANGED to keep the existing setting.
m_groupNum	short	Specifies the group number or PEP_UNCHANGED to keep the existing group number.

m_mutuallyExclusiveGroup	short	Specifies whether or not the group assigned to a field in a report is mutually exclusive. This variable does not apply if the m_partOfGroup member variable is set to FALSE. Set this variable to TRUE if it is mutually exclusive, FALSE if it is not, or PEP_UNCHANGED to keep the existing setting.

Constructor CRPEParameterValueInfo::CRPEParameterValueInfo

This constructs a CRPEParameterValueInfo 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.

Constructor Syntax (Default)

```
CRPEPrameterValueInfo ();
```

Constructor Syntax

```
CRPEParameterValueInfo (
          short isNullable,
          short disallowEditing,
          short allowMultipleValues,
          short hasDiscreteValues,
          short partOfGroup,
          short groupNum,
          short mutuallyExclusiveGroup );
```

CRPEPrintOptions

This structure contains print options for a report and is used by *CRPEJob::SetPrintOptions, Page 96*.

Data Members

Each data member is set by the corresponding constructor parameter unless otherwise noted.

Member	*Type*	*Default*	*Description*
m_startPageN	unsigned short	0	Specifies the first page that you want to print. Parameter is 1-based. Use 0 to preserve the existing setting.
m_stopPageN	unsigned short	0	Specifies the last page that you want to print. Parameter is 1-based. Use 0 to preserve the existing setting.
m_nReportCopies	unsigned short	0	Specifies the number of copies that you want to print. Parameter is 1-based. Use 0 to preserve the existing setting.

Member	*Type*	*Default*	*Description*
m_collation	unsigned short	PEP_ DEFAULT COLLATION	Indicates whether you want the copies of the report to be collated (if you are printing multiple copies of a multiple page report). Uses one of the following constants.
	Constant		***Description***
	PEP_UNCOLLATED		Page order = 1, 1, 1,...; 2, 2, 2,...; 3, 3, 3,...; etc.
	PEP_COLLATED		Page order = 1, 2, 3,...; 1, 2, 3,...; 1, 2, 3,...; etc.
	PEP_DEFAULTCOLLATION		Use the collation settings specified in the report.
Member	***Type***	***Default***	***Description***
m_outputFileName	_TCHAR	'\0'	Specifies the output file name.

Constructor CRPEPrintOptions::CRPEPrintOptions

This 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.

Constructor Syntax (Default)

```
CRPEPrintOptions ();
```

Constructor Syntax

```
CRPEPrintOptions (
          unsigned short startPageN,
          unsigned short stopPageN,
          unsigned short nReportCopies,
          unsigned short collation,
          _TCHAR *outputFileName );
```

CRPEReadingRecordsEventInfo

This structure provides information about records read when a callback function is called with event ID equal to PEP_READING_RECORDS_EVENT.

Data Members

Member	*Type*	*Default*	*Description*
m_cancelled	short	FALSE	BOOL. Indicates whether the reading records event is canceled.
m_recordsRead	long	0	Indicates how many records have been read so far.

Member	*Type*	*Default*	*Description*
m_recordsSelected	long	0	Indicates how many records have been selected so far.
m_done	short	FALSE	BOOL. Indicates whether the reading records event is finished.

Constructor CRPEReadingRecordsEventInfo::CRPEReadingRecordsEventInfo

This constructs a CRPEReadingRecordsEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEReadingRecordsEventInfo ();
```

CRPEReportFieldMappingInfo

This structure contains report field mapping information.

Data Members

Each data member is set by the corresponding constructor parameter unless otherwise noted.

Data Member	*Type*	*Default*	*Description*
m_valueType	WORD	0	valueType, one of the PEP_FVT constants.

Constant
PEP_FVT_INT8SFIELD
PEP_FVT_INT8UFIELD
PEP_FVT_INT16SFIELD
PEP_FVT_INT16UFIELD
PEP_FVT_INT32SFIELD
PEP_FVT_INT32UFIELD
PEP_FVT_NUMBERFIELD
PEP_FVT_CURRENCYFIELD
PEP_FVT_BOOLEANFIELD
PEP_FVT_DATEFIELD
PEP_FVT_TIMEFIELD
PEP_FVT_STRINGFIELD
PEP_FVT_TRANSIENTMEMOFIELD
PEP_FVT_PERSISTENTMEMOFIELD
PEP_FVT_BLOBFIELD

	PEP_FVT_DATETIMEFIELD
	PEP_FVT_BITMAPFIELD
	PEP_FVT_ICONFIELD
	PEP_FVT_PICTUREFIELD
	PEP_FVT_OLEFIELD
	PEP_FVT_GRAPHFIELD
	PEP_FVT_UNKNOWNFIELD

Data Member	*Type*	*Default*	*Description*
m_TableAliasName	_TCHAR	'\0'	The table alias name (of length PEP_TABLE_NAME_LEN = 128).
m_databaseFieldName	_TCHAR	'\0'	The database field name (of length PEP_DATABASE_FIELD_NAME_LEN = 128).
m_mappingTo	int	0	The assigned index of a field in PEFieldMappingEventInfo. See Remarks below.

Remarks

- Mapped fields are assigned to the index of a field in the member databaseFields array of PEFieldMappingEventInfo.
- Unmapped fields are assigned to -1.

Constructor CRPEReportFieldMappingInfo::CRPEReportFieldMappingInfo

This constructs a CRPEReportFieldMappingInfo 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.

Constructor Syntax (Default)

```
CRPEReportFieldMappingInfo ();
```

Constructor Syntax

```
CRPEReportFieldMappingInfo (
          WORD valueType,
          const _TCHAR *tableAliasName,
          const _TCHAR *databaseFieldName,
          int  mappingTo );
```

CRPEReportOptions

This structure contains specifications for formatting selected report sections. This information is used by the methods *CRPEJob::GetReportOptions, Page 61*, and *CRPEJob::SetReportOptions, Page 96*.

Data Members

- Each data member is set by the corresponding constructor parameter.
- Use TRUE, FALSE, or PEP_UNCHANGED for each member except as noted below.
- The default for each data member is PEP_UNCHANGED.

m_saveDataWithReport	Specifies whether data should be saved with the report.
m_saveSummariesWithReport	Specifies whether to save summaries with the report.
m_useIndexForSpeed	Specifies whether to use index values.
m_translateDOSStrings	Specifies whether to translate DOS strings.
m_translateDOSMemos	Specifies whether to translate DOS memos.
m_convertDateTimeType	Specifies whether to convert Date/Time format to another format. Use one of the following constants or PEP_UNCHANGED.

Constant	***Description***
PEP_RPTOPT_DVTDATETIMETODATE	To Date.
PEP_RPTOPT_KEEPDATETIMETYPE	To Date/Time.

m_convertNullFieldToDefault	Specifies whether to convert NULL parameter fields to their default values.
m_morePrintEngineErrorMessages	Specifies whether to allow the print engine to pop up error messages to the screen from an application without the user's intervention.
m_caseInsensitiveSQLData	Specifies whether to perform a case Insensitive search for SQL data.
m_verifyOnEveryPrint	Specifies whether to perform database verification for every print job.
m_zoomMode	Use one of the PEP_ZOOM_XXX **Zoom Constants** (page 176) or PEP_UNCHANGED.
m_hasGroupTree	Specifies whether there is a group tree associated with the report.
m_dontGenerateDataForHiddenObjects	Specifies whether to generate data for hidden objects.
m_performGroupingOnServer	Specifies whether to perform grouping on servers.
m_doAsyncQuery	Specifies whether to do Async Query.
m_promptMode	Specifies the prompt mode. Uses **Report Prompt Option Constants** (page 174).
m_selectDistinctRecords	Boolean. Specifies the select distinct records option.
m_wysiwygMode	Boolean. Specifies whether to use wysiwyg mode.

Constructor CRPEReportOptions::CRPEReportOptions

This constructs a CRPEReportOptions 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.

Constructor Syntax (Default)

```
CRPEReportOptions ();
```

Constructor Syntax

```
CRPEReportOptions (
          short saveDataWithReport,
          short saveSummariesWithReport,
          short useIndexForSpeed,
          short translateDOSStrings,
          short translateDOSMemos,
          short convertDateTimeType,
          short convertNullFieldToDefault,
          short morePrintEngineErrorMessages,
          short caseInsensitiveSQLData,
          short verifyOnEveryPrint,
          short zoomMode,
          short hasGroupTree,
          short dontGenerateDataForHiddenObjects,
          short performGroupingOnServer,
          short doAsyncQuery,
          short promptMode,
          short selectDistinctRecords,
          short wysiwygMode );
```

CRPEReportSummaryInfo

This structure provides report summary information such as the report title, author, and comments. It corresponds to the report summary information in the Seagate Crystal Reports Designer. This structure is used by *CRPEJob::GetReportSummaryInfo, Page 61,* to get information about a specific parameter field and by *CRPEJob::SetReportSummaryInfo, Page 97,* to change a specific parameter field.

Data Members

- Each data member is set by the corresponding constructor parameter unless otherwise noted.
- The default value for each data member is '\0', except as noted.

m_applicationName	_TCHAR	Specifies an application name (of size PEP_SI_APPLICATION_NAME_LEN = 128) for the application using this report.

m_title	_TCHAR	Specifies the title (of size PEP_SI_TITLE_LEN = 128) of the current report.
m_subject	_TCHAR	Specifies the subject (of size PEP_SI_SUBJECT_LEN = 128) of the current report.
m_author	_TCHAR	Specifies the author (of size PEP_SI_AUTHOR_LEN = 128) of the current report.
m_keywords	_TCHAR	Specifies the keywords (of size PEP_SI_KEYWORDS_LEN = 128) included for the current report.
m_comments	_TCHAR	Specifies any comments (of size PEP_SI_COMMENTS_LEN = 512) for the current report.
m_reportTemplate	_TCHAR	Specifies the report template (of size PEP_SI_REPORT_TEMPLATE_LEN = 128) for the current report.
m_savePreview Picture	short	Specifies whether or not to save the Preview Picture. Default value = 0.

Constructor CRPEReportSummaryInfo:: CRPEReportSummaryInfo

This constructs a CRPEReportSummaryInfo 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.

Constructor Syntax (Default)

```
CRPEReportSummaryInfo ();
```

Constructor Syntax

```
CRPEReportSummaryInfo (
          _TCHAR *applicationName,
          _TCHAR *title,
          _TCHAR *subject,
          _TCHAR *author,
          _TCHAR *keywords,
          _TCHAR *comments,
          _TCHAR *reportTemplate,
          short savePreviewPicture );
```

CRPESearchButtonClickedEventInfo

This structure returns information when a callback function is called with event ID equal to PEP_SEARCH_BUTTON_CLICKED_EVENT.

Data Members

Member	*Type*	*Default*	*Description*
m_windowHandle	long	0	Specifies on which frame window handle the button is positioned.
m_searchString	_TCHAR	'\0'	Specifies the search string (of length = PEP_SEARCH_STRING_LEN = 128) in search edit control.

Constructor CRPESearchButtonClickedEventInfo::CRPESearchButtonClickedEventInfo

This constructs a CRPESearchButtonClickedEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPESearchButtonClickedEventInfo ();
```

CRPESectionOptions

This structure contains specifications for formatting selected report sections and areas.

Data Members

- Each data member is set by the corresponding constructor parameter.
- The default for each member is FALSE except m_backgroundColour default PEP_UNCHANGED_COLOR.

m_visible	short	Specifies whether or not the selected section is visible. Use TRUE to keep the section visible, FALSE to hide the section, or PEP_UNCHANGED to retain the previous setting.
m_newPageBefore	short	Specifies whether or not the Seagate Crystal Report Engine will insert a page break before the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break, or PEP_UNCHANGED to retain the previous setting.
m_newPageAfter	short	Specifies whether or not the Seagate Crystal Report Engine will insert a page break after the section is printed. Use TRUE to insert a page break, FALSE to leave it without a page break, or PEP_UNCHANGED to retain the previous setting.

m_keepTogether	short	Specifies whether the Seagate 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. Use TRUE to keep the lines together, FALSE to allow the Seagate Crystal Report Engine to split section data from one page to the next if necessary, or PEP_UNCHANGED to retain the previous setting.
m_suppressBlank Lines	short	Specifies whether the Seagate 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, or PEP_UNCHANGED retains the previous setting.
m_resetPageNAfter	short	Specifies whether the Seagate 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 continue the existing numbering, or PEP_UNCHANGED to retain the previous setting.
m_printAtBottomOf Page	short	Specifies whether the Seagate 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, or PEP_UNCHANGED to retain the previous setting.
m_background Colour	COLOREF	Specifies the RGB color value contained in the structure, **COLORREF** (*Seagate Crystal Reports Technical Reference Guide*). Use PEP_UNCHANGED_COLOR to retain the previous settings.
m_underlaySection	short	Indicates whether the specified section is to use the underlay feature for chart and map placement. Use TRUE to underlay. Use PEP_UNCHANGED to retain the previous setting.
m_showArea	short	Indicates whether to show/hide the Area to which the specified section belongs. Use TRUE to show the specified Area, FALSE to hide it, or PEP_UNCHANGED to retain the previous setting.
m_freeForm Placement	short	Design time flag. If set to TRUE an object can be placed anywhere in a section. Use PEP_UNCHANGED to retain the previous setting.
m_reserveMinimum PageFooter	short	Used to reduce unnecessary white space in the page footer area containing more than one conditionally formatted section . When set to TRUE, the space required to display only one section (the tallest) is reserved. When set to FALSE (default), the maximum height necessary to display every section in the page footer area at full height will be reserved. See Remarks below.

Remarks

reserveMinimumPageFooter can be used to remove undesirable white space when there are two or more sections in the page footer area and only one section, based on conditional settings, will be visible when the report is viewed in the preview window. For example, a report page footer area might contain two sections,

one conditionally set to display at the bottom of odd pages and the other at the bottom of even pages. When this member is set to TRUE, sufficient space will be reserved to display only one of the sections included in the area (set to the height of the tallest section). Note that when reserveMinimumPageFooter = TRUE and there is more than one section visible in the page footer area, the visible sections will be displayed only to the extent allowed by the reserved minimum space (the height of the tallest section) with the remainder truncated.

Constructor CRPESectionOptions::CRPESectionOptions

This 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.

Constructor Syntax (Default)

```
CRPESectionOptions ();
```

Constructor Syntax

```
CRPESectionOptions (
          short visible,
          short newPageBefore,
          short newPageAfter,
          short keepTogether,
          short suppressBlankLines,
          short resetPageNAfter,
          short printAtBottomOfPage,
          COLORREF backgroundColour,
          short underlaySection,
          short showArea,
          short freeFormPlacement,
          short reserveMinimumPageFooter );
```

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 57,* and *CRPEJob::SetNthTableSessionInfo, Page 93,* member functions.

Data Members

Member	*Type*	*Default*	*Description*
m_userID	_TCHAR	'\0'	Specifies the user ID (of length PEP_SESS_USERID_LEN = 128) needed for logging on to the MS Access system (NULL-terminated).

Member	*Type*	*Default*	*Description*
m_password	_TCHAR	'\0'	Specifies the password (of length PEP_SESS_ PASSWORD_LEN = 128) needed for logging on to the MS Access system (NULL-terminated).
m_sessionHandle	DWORD	0	The handle to the current MS Access session. When getting information from a report, sessionHandle is undefined. When setting information for a report, if sessionHandle is = 0 then the UserID and Password settings are used; otherwise the SessionHandle is used.

Constructor CRPESessionInfo::CRPESessionInfo

This 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.

Constructor Syntax (Default)

```
CRPESessionInfo ();
```

Constructor Syntax

```
CRPESessionInfo (
          const _TCHAR *userID,
          const _TCHAR *password,
          DWORD sessionHandle );
```

CRPEShowGroupEventInfo

This structure provides show group event information when an event callback is called with event ID equal to PEP_SHOW_GROUP_EVENT.

Data Members

Member	*Type*	*Default*	*Description*
m_groupLevel	WORD	0	Number of items in groupList.
m_windowHandle	long	0	Frame window handle where the event happens.
m_groupList	_TCHAR	NULL	Pointer to an array of group names describing the group path that the group area is going to navigate to. This pointer is freed after the callback function.

Constructor CRPEShowGroupEventInfo::CRPEShowGroupEventInfo

This constructs a CRPEShowGroupEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEShowGroupEventInfo ();
```

CRPEStartEventInfo

This structure provides start event information when the callback function is called with event ID equal to PEP_START_EVENT.

Data Members

The default for data member m_destination is PEP_TO_NOWHERE.

m_destination	WORD	Indicates the process destination using one of the following PEP_TO_XXX **Job Destination Constants** (page 173).

Constructor CRPEStartEventInfo::CRPEStartEventInfo

This constructs a CRPEStartEventInfo structure object. It assigns the default value to the member of the structure.

Constructor Syntax

```
CRPEStartEventInfo ();
```

CRPEStopEventInfo

This structure provides stop event information when a callback method is called with an event ID equal to PEP_STOP_EVENT.

Data Members

m_destination	WORD	Indicates the process destination using one of the PEP_TO_XXX **Job Destination Constants** (page 173).
m_jobStatus	WORD	Indicates the current status of the job using one of the PEP_JOBXXX **Job Status Constants** (page 173).

Constructor CRPEStopEventInfo::CRPEStopEventInfo

This constructs a CRPEStopEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEStopEventInfo ();
```

CRPESubreportInfo

The CRPESubreportInfo structure contains information on a subreport that appears in the main report. This structure is used by *CRPEJob::GetSubreportInfo, Page 66*, to retrieve information about a subreport.

Data Members

Member	*Type*	*Default*	*Description*
m_name	_TCHAR	'\0'	A string (of length PEP_SUBREPORT_NAME_LEN = 128) that contains the specified subreport name.
m_NLinks	short	0	Indicates the number of links that are present in the specified subreport.
m_IsOnDemand	short	FALSE	Indicates whether the specified subreport is available to the user on demand. Set to TRUE if the subreport is available on demand, FALSE if it is not, and PEP_UNCHANGED to retain the previous setting.
m_external	short	FALSE	Specifies TRUE if the subreport is imported and FALSE otherwise.
m_reimportOption	short	PEP_UNCHANGED	Specifies the reimport option using one of the PEP_SRI_XXX **Subreport Info Constants** (page 175).

Constructor CRPESubreportInfo::CRPESubreportInfo

This constructs a CRPESubreportInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPESubreportInfo();
```

CRPETableDifferenceInfo

This structure contains table difference information and is used by *CRPEJob::CheckNthTableDifferences, Page 21*.

Data Member

Each data member is set by the corresponding constructor parameter unless otherwise noted.

Member	*Type*	*Default*	*Description*
m_tableDifferences	DWORD	0	Contains any combination of PEP_TCD_XXX **Table Difference Constants** (page 175).
m_reserved1	DWORD		Reserved. Do not use.
m_reserved2	DWORD		Reserved. Do not use.

Constructor CRPETableDifferenceInfo::CRPETableDifferenceInfo

This constructs a CRPETableDifferenceInfo 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.

Constructor Syntax (Default)

```
CRPETableDifferenceInfo ();
```

Constructor Syntax

```
CRPETableDifferenceInfo (
          WORD StructSize,
          DWORD tableDifferences );
```

CRPETableLocation

This structure contains table location information and is used by *CRPEJob::GetNthTableLocation, Page 55*, and *CRPEJob::SetNthTableLocation, Page 91*.

Data Member

Each data member is set by the corresponding constructor parameter unless otherwise noted.

Member	*Type*	*Default*	*Description*
m_location	_TCHAR	'\0'	Specifies a pointer to the string (of length PEP_TABLE_LOCATION_LEN = 256) that contains the database location (NULL-terminated).
m_SubLocation	_TCHAR	'\0'	Specifies a pointer to the string (of length PEP_TABLE_LOCATION_LEN = 256) that contains the database sublocation (NULL-terminated).
m_ConnectBuffer	_TCHAR	'\0'	Specifies a pointer to the string (of length PEP_CONNECTION_BUFFER_LEN = 512) that contains the database connection buffer (NULL-terminated).

Constructor CRPETableLocation::CRPETableLocation

This 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.

Constructor Syntax (Default)

```
CRPETableLocation ();
```

Constructor Syntax

```
CRPETableLocation (
          const _TCHAR *location,
          const _TCHAR *SubLocation,
          const _TCHAR *ConnectionBuffer );
```

CRPETablePrivateInfo

This structure contains information for using data objects such as ADO, RDO, or CDO with the Active Data Driver.

Data Member

Each data member is set by the corresponding constructor parameter unless otherwise noted.

Member	*Type*	*Description*
m_nBytes	WORD	Specifies the length of the data starting at the dataPtr.
m_tag	DWORD	Specifies a value indicating the type of data being passed to theDatabaseTable 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.
m_dataPtr	BYTE	Specifies a pointer to variant data passed to the database driver. With Active data, this must be a Recordset object if DAO, ADO, or the Visual Basic data control is used. If CDO is used, this must be a Rowset object.

Constructor CRPETablePrivateInfo::CRPETablePrivateInfo

This constructs a CRPETablePrivateInfo 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.

Constructor Syntax (Default)

```
CRPETablePrivateInfo ();
```

Constructor Syntax

```
CRPETablePrivateInfo (
          WORD StructSize,
          WORD nBytes,
          DWORD tag,
          BYTE *dataPtr );
```

CRPETableType

This structure contains information about a specific table used in the report and is used by *CRPEJob::GetNthTableSessionInfo, Page 57*.

Data Members

Member	*Type*	*Default*	*Description*
m_dllName	_TCHAR	'\0'	Specifies the name of the appropriate database DLL (of length PEP_DLL_NAME = 64) for the table of interest (NULL-terminated).
m_descriptiveName	_TCHAR	'\0'	Specifies the name of the database table (of length PEP_FULL_NAME = 256) of interest (NULL-terminated).
m_dbType	WORD	PEP_DT_SQL	Specifies the type of database that contains the table of interest. Use one of the following constants.
	Constant		***Description***
	PEP_DT_STANDARD		Standard data table.
	PEP_DT_SQL		SQL data table.

Constructor CRPETableType::CRPETableType

This constructs a CRPETableType structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPETableType ();
```

CRPETrackCursorInfo

This structure specifies cursor information such as size and type and is used by *CRPEJob::GetTrackCursorInfo, Page 67*, or *CRPEJob::SetTrackCursorInfo, Page 101*.

Data Members

- Each available data member uses the PEP_TC_XXX **Cursor Constants** (page 158), except as noted below.
- The default value for each available data member is PEP_TC_DEFAULT_CURSOR, except as noted below. For most areas, CRPE sets this to PEP_TC_ARROW_CURSOR.

- Pass PE_UNCHANGED for any available data member to use the existing cursor.

Member	*Type*	*Description*
m_groupAreaCursor:	short	Specifies the group area cursor.
m_groupAreaFieldCursor	short	Specifies the cursor for a memo, blob, database, summary, or formula field in the group area.
m_detailAreaCursor	short	Specifies the cursor for the Details area.
m_detailAreaFieldCursor	short	Specifies the cursor for a memo, blob, database, summary, or formula field in the Details area.
m_graphCursor	short	Specifies the cursor for the group chart in the Report Header or Report Footer area.
m_groupAreaCursorHandle	long	Reserved, do not use.
m_groupAreaFieldCursorHandle	long	Reserved, do not use.
m_detailAreaCursorHandle	long	Reserved, do not use.
m_detailAreaFieldCursorHandle	long	Reserved, do not use.
m_graphCursorHandle	long	Reserved, do not use.
m_ondemandSubreportCursor	short	Specifies the cursor for real-time subreports when drilldown for the window is enabled. Default is PEP_TC_MAGNIFY_CURSOR.
m_hyperlinkCursor	short	Specifies the cursor to show over a report object that has hyperlink text. Default is PEP_TC_HAND_CURSOR.

Remarks

- By default, most of the cursors are set to the constant PEP_TC_ARROW_CURSOR. If the canDrillDown option in **CRPEWindowOptions** (page 153) is set to True, groupAreaCursor, groupAreaFieldCursor, and graphCursor will be PEP_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.

Constructor CRPETrackCursorInfo::CRPETrackCursorInfo

This constructs a RPETrackCursorInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPETrackCursorInfo () ;
```

CRPEValueInfo

This structure contains parameter field value information.

Data Members

Each data member is set by the corresponding constructor parameter.

Member	*Type*	*Default*	*Description*
m_valueType	WORD	PEP_VI_ NUMBER	Specifies the data type of the parameter field. The Seagate Crystal Report Engine supports the following data types and associated constants.
		Data Type	***Constant***
		Number	PEP_VI_NUMBER
		Currency	PEP_VI_CURRENCY
		Boolean	PEP_VI_BOOLEAN
		Date	PEP_VI_DATE
		String	PEP_VI_STRING
		Date/Time	PEP_VI_DATETIME
		Time	PEP_VI_TIME
		Integer	PEP_VI_INTEGER
		Color	PEP_VI_COLOR
		Char	PEP_VI_CHAR
		Long	PEP_VI_LONG
		NoValue	PEP_VI_NOVALUE
Member	***Type***	***Default***	***Description***
m_viNumber	double	0	Specifies the value if the parameter is a numeric value.
m_viCurrency	double	0	Specifies the value if the parameter is a currency value.
m_viBoolean	BOOL	FALSE	Specifies the value if the parameter is a Boolean value.
m_viString	_TCHAR	\0’	Specifies the string (of length PEP_VI_STRING_LEN = 256, NULL-terminated) value if the parameter is a string
m_viDate	short	0, 0, 0	Specifies the value if the parameter is a date value (year, month, day).
m_viDateTime	short	0, 0, 0, 0, 0, 0	Specifies the value if the parameter is a date/time value (year, month, day, hour, minute, second).

m_viTime	short	0, 0, 0	Specifies the value if the parameter is a time value (hour, minute, second).
m_viColor	COLOREF	0X00000000	For future support of color parameters.
m_viInteger	short	0	For future support of integer parameters.
m_viC	_TCHAR	'\0'	For future support of character parameters.
m_ignored	_TCHAR	'\0'	For 4 byte alignment. Ignored. Do not use.
m_viLong	long	0	For future support of long parameters.

Constructor CRPEValueInfo::CRPEValueInfo

This constructs a CRPEValueInfo 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.

Constructor Syntax (Default)

```
CRPEValueInfo ();
```

Constructor Syntax

```
CRPEValueInfo (
          WORD StructSize,
          WORD valueType,
          double viNumber,
          double viCurrency,
          BOOL viBoolean,
          _TCHAR viString,
          short viDate,
          short viDateTime,
          short viTime,
          COLORREF viColor,
          short viInteger,
          _TCHAR viC,
          _TCHAR ignored,
          long viLong );
```

CRPEWindowOptions

This structure is used to set attributes for a print window. Use *CRPEJob::GetWindowOptions, Page 67*, and *CRPEJob::SetWindowOptions, Page 101*, to retrieve and change these options.

Data Members

- The default for each data member is FALSE.
- For each data member, use TRUE, FALSE, or PEP_UNCHANGED to use existing setting.

m_hasGroupTree	short	Indicates whether the specified print window has a group tree.
m_canDrillDown	short	Indicates whether the specified print window has permission to drill down.
m_has NavigationControls	short	Indicates whether the specified print window has navigation controls (previous, next, etc.).
m_hasCancelButtom	short	Indicates whether the specified print window has a Cancel button.
m_hasPrintButton	short	Indicates whether the specified print window has a Print button.
m_hasExportButton	short	Indicates whether the specified print window has an Export button.
m_has ZoomControl	short	Indicates whether the specified print window has magnification controls.
m_hasCloseButton	short	Indicates whether the specified print window has a Close button.
m_has progressControls	short	Indicates whether the specified print window has controls to indicated the progress of a lengthy operation.
m_hasSearchButton	short	Indicates whether to perform database verification for every print job.
m_hasPrintSetupButton	short	Indicates whether the specified print window has a Print Setup button.
m_hasRefreshButton	short	Indicates whether the specified print window has a Refresh Data button.
m_showToolbarTips	short	Indicates whether Toolbar Tips will be shown or hidden.
m_showDocumentTips	short	Indicates whether Document Tips will be shown or hidden.
m_hasLaunchButton	short	Indicates whether a Launch Seagate Analysis button will be shown or hidden.

Constructor CRPEWindowOptions::CRPEWindowOptions

This constructs a CRPEWindowOptions 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.

Constructor Syntax (Default)

```
CRPEReportOptions ();
```

Constructor Syntax

```
CRPEWindowOptions (
          short hasGroupTree,
          short canDrillDown,
          short hasNavigationControls,
          short hasCancelButton,
          short hasPrintButton,
          short hasExportButton,
          short hasZoomControl,
          short hasCloseButton,
          short hasProgressControls,
          short hasSearchButton,
          short hasPrintSetupButton,
          short hasRefreshButton
          short showToolbarTips,
          short showDocumentTips,
          short hasLaunchButton );
```

CRPEZoomLevelChangingEventInfo

This structure gives information about the PEP_ZOOM_LEVEL_CHANGING_EVENT event.

Data Members

m_zoomLevel	WORD	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 PEP_ZOOM_XXX **Zoom Constants** (page 176).
m_windowHandle	long	Specifies the frame window handle where the event happens.

Constructor CRPEZoomLevelChangingEventInfo::CRPEZoomLevelChangingEventInfo

This constructs a CRPEZoomLevelChangingEventInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEZoomLevelChangingEventInfo ();
```

class CRPEngine and class CRPEJob Constants

This section describes some of the more widely used class CRPEngine and class CRPEJob constants.

Area/Section Format Formula Constants

Constants PE_FFN_SECTION_VISIBILITY and PEP_FFN_SECTION_BACK_COLOUR are included for support of older applications. These constants have the same value as PE_FFN_AREASECTION_VISIBILITY and PEP_FFN_SECTION_BACK_COLOR. All new Seagate Crystal Reports applications should use PE_FFN_AREASECTION_VISIBILITY and PEP_FFN_SECTION_BACK_COLOR.

Constant	***Description***
PEP_FFN_AREASECTION_VISIBILITY	Area and Section format
PEP_FFN_SECTION_VISIBILITY	Section format
PEP_FFN_SHOW_AREA	Area format
PEP_FFN_NEW_PAGE_BEFORE	Area and Section format
PEP_FFN_NEW_PAGE_AFTER	Area and Section format
PEP_FFN_KEEP_TOGETHER	Area and Section format
PEP_FFN_SUPPRESS_BLANK_SECTION	Section format
PEP_FFN_RESET_PAGE_N_AFTER	Area and Section format
PEP_FFN_PRINT_AT_BOTTOM_OF_PAGE	Area and Section format

Constant	*Description*
PEP_FFN_UNDERLAY_SECTION	Section format
PEP_FFN_SECTION_BACK_COLOUR	Section format
PEP_FFN_SECTION_BACK_COLOR	Section format

Cursor Constants

Constant	*Value*	*Description*
PEP_TC_DEFAULT_CURSOR	0	CRPE set default cursor to be PEP_TC_ARROW_CURSOR
PEP_TC_ARROW_CURSOR	1	CRPE default cursor
PEP_TC_CROSS_CURSOR	2	
PEP_TC_IBEAM_CURSOR	3	
PEP_TC_UPARROW_CURSOR	4	
PEP_TC_SIZEALL_CURSOR	5	
PEP_TC_SIZENWSE_CURSOR	6	
PEP_TC_SIZENESW_CURSOR	7	
PEP_TC_SIZEWE_CURSOR	8	
PEP_TC_SIZENS_CURSOR	9	
PEP_TC_NO_CURSOR	10	
PEP_TC_WAIT_CURSOR	11	
PEP_TC_APPSTARTING_CURSOR	12	
PEP_TC_HELP_CURSOR	13	
PEP_TC_SIZE_CURSOR	14	For 16bit
PEP_C_ICON_CURSOR	15	For 16bit
PEP_TC_BACKGROUND_PROCESS_CURSOR	94	CRPE specific cursor.
PEP_TC_GRAB_HAND_CURSOR	95	CRPE specific cursor.
PEP_TC_ZOOM_IN_CURSOR	96	CRPE specific cursor.
PEP_TC_REPORT_SECTION_CURSOR	97	CRPE specific cursor.
PEP_TC_HAND_CURSOR	98	CRPE specific cursor.
PEP_TC_MAGNIFY_CURSOR	99	CRPE specific cursor.

Error Codes

The following codes are returned by *CRPEngine::GetErrorCode, Page 6*, and *CRPEJob::GetErrorCode, Page 30*.

Constant	*Description*
PEP_ERR_NOERROR	
PEP_ERR_NOTENOUGHMEMORY	
PEP_ERR_INVALIDJOBNO	
PEP_ERR_INVALIDHANDLE	
PEP_ERR_STRINGTOOLONG	
PEP_ERR_NOSUCHREPORT	
PEP_ERR_NODESTINATION	
PEP_ERR_BADFILENUMBER	
PEP_ERR_BADFILENAME	
PEP_ERR_BADFIELDNUMBER	
PEP_ERR_BADFIELDNAME	
PEP_ERR_BADFORMULANAME	
PEP_ERR_BADSORTDIRECTION	
PEP_ERR_ENGINENOTOPEN	
PEP_ERR_INVALIDPRINTER	
PEP_ERR_PRINTFILEEXISTS	
PEP_ERR_BADFORMULATEXT	
PEP_ERR_BADGROUPSECTION	
PEP_ERR_ENGINEBUSY	
PEP_ERR_BADSECTION	
PEP_ERR_NOPRINTWINDOW	
PEP_ERR_JOBALREADYSTARTED	
PEP_ERR_BADSUMMARYFIELD	
PEP_ERR_NOTENOUGHSYSRES	
PEP_ERR_BADGROUPCONDITION	
PEP_ERR_JOBBUSY	
PEP_ERR_BADREPORTFILE	
PEP_ERR_NODEFAULTPRINTER	
PEP_ERR_SQLSERVERERROR	
PEP_ERR_BADLINENUMBER	
PEP_ERR_DISKFULL	

Constant	*Description*
PEP_ERR_FILEERROR	
PEP_ERR_INCORRECTPASSWORD	
PEP_ERR_BADDATABASEDLL	
PEP_ERR_BADDATABASEFILE	
PEP_ERR_ERRORINDATABASEDLL	
PEP_ERR_DATABASESESSION	
PEP_ERR_DATABASELOGON	
PEP_ERR_DATABASELOCATION	
PEP_ERR_BADSTRUCTSIZE	
PEP_ERR_BADDATE	
PEP_ERR_BADEXPORTDLL	
PEP_ERR_ERRORINEXPORTDLL	
PEP_ERR_PREVATFIRSTPAGE	
PEP_ERR_NEXTATLASTPAGE	
PEP_ERR_CANNOTACCESSREPORT	
PEP_ERR_USERCANCELLED	
PEP_ERR_OLE2NOTLOADED	
PEP_ERR_BADCROSSTABGROUP	
PEP_ERR_NOCTSUMMARIZEDFIELD	
PEP_ERR_DESTINATIONNOTEXPORT	
PEP_ERR_INVALIDPAGENUMBER	
PEP_ERR_NOTSTOREDPROCEDURE	
PEP_ERR_INVALIDPARAMETER	
PEP_ERR_GRAPHNOTFOUND	
PEP_ERR_INVALIDGRAPHTYPE	
PEP_ERR_INVALIDGRAPHDATA	
PEP_ERR_CANNOTMOVEGRAPH	
PEP_ERR_INVALIDGRAPHTEXT	
PEP_ERR_INVALIDGRAPHOPT	
PEP_ERR_BADSECTIONHEIGHT	
PEP_ERR_BADVALUETYPE	
PEP_ERR_INVALIDSUBREPORTNAME	
PEP_ERR_NOPARENTWINDOW	No dialog parent window.
PEP_ERR_INVALIDZOOMFACTOR	Invalid zoom factor.

Constant	*Description*
PEP_ERR_PAGESIZEOVERFLOW	
PEP_ERR_LOWSYSTEMRESOURCES	
PEP_ERR_BADGROUPNUMBER	
PEP_ERR_INVALIDOBJECTFORMATNAME	
PEP_ERR_INVALIDNEGATIVEVALUE	
PEP_ERR_INVALIDMEMORYPOINTER	
PEP_ERR_INVALIDOBJECTTYPE	
PEP_ERR_INVALIDGRAPHDATATYPE	
PEP_ERR_INVALIDSUBREPORTLINKNUMBER	
PEP_ERR_SUBREPORTLINKEXIST	
PEP_ERR_BADROWCOLVALUE	
PEP_ERR_INVALIDSUMMARYNUMBER	
PEP_ERR_INVALIDGRAPHDATAFIELDNUMBER	
PEP_ERR_INVALIDSUBREPORTNUMBER	
PEP_ERR_INVALIDFIELDSCOPE	
PEP_ERR_FIELDINUSE	
PEP_ERR_INVALIDPARAMETERNUMBER	
PEP_ERR_INVALIDPAGEMARGINS	
PEP_ERR_REPORTONSECUREQUERY	
PEP_ERR_CANNOTOPENSECUREQUERY	
PEP_ERR_INVALIDSECTIONNUMBER	
PEP_ERR_SQLSERVERNOTOPENED	
PEP_ERR_TABLENAMEEXIST	
PEP_ERR_INVALIDCURSOR	
PEP_ERR_FIRSTPASSNOTFINISHED	
PEP_ERR_CREATEDATASOURCE	
PEP_ERR_CREATEDRILLDOWNPARAMETERS	
PEP_ERR_CHECKFORDATASOURCECHANGES	
PEP_ERR_STARTBACKGROUNDPROCESSING	
PEP_ERR_SQLSERVERINUSE	
PEP_ERR_GROUPSORTFIELDNOTSET	
PEP_ERR_CANNOTSETSAVESUMMARIES	
PEP_ERR_LOADOLAPDATABASEMANAGER	
PEP_ERR_OPENOLAPCUBE	

Constant	*Description*
PEP_ERR_READOLAPCUBEDATA	
PEP_ERR_CANNOTSAVEQUERY	
PEP_ERR_CANNOTREADQUERYDATA	
PEP_ERR_MAINREPORTFIELDLINKED	
PEP_ERR_INVALIDMAPPINGTYPEVALUE	
PEP_ERR_HITTESTFAILED	
PEP_ERR_BADSQLEXPRESSIONNAME	No SQL expression by the specified *name* exists in this report.
PEP_ERR_BADSQLEXPRESSIONNUMBER	No SQL expression by the specified *number* exists in this report.
PEP_ERR_BADSQLEXPRESSIONTEXT	Not a valid SQL expression.
PEP_ERR_INVALIDDEFAULTVALUEINDEX	Invalid index for default value of a parameter.
PEP_ERR_NOMINMAXVALUE	The specified PEP_PF_XXX type does not have min/max values.
PEP_ERR_INCONSISTANTTYPES	Both min and max values are specified in PESetParameterMinMaxValue and the value types for the min and max values are not the same.
PEP_ERR_CANNOTLINKTABLES	
PEP_ERR_CREATEROUTER	
PEP_ERR_INVALIDFIELDINDEX	
PEP_ERR_INVALIDGRAPHTITLETYPE	
PEP_ERR_INVALIDGRAPHTITLEFONTTYPE	
PEP_ERR_PARAMTYPEDIFFERENT	The type used in an add/set value API for a parameter differs with its existing type.
PEP_ERR_INCONSISTANTRANGETYPES	The value type for both start & end range values must be the same.
PEP_ERR_RANGEORDISCRETE	An operation was attempted on a discrete parameter that is only legal for range parameters or vice versa.
PEP_ERR_NOTMAINREPORT	An operation was attempted that is disallowed for subreports.
PEP_ERR_INVALIDCURRENTVALUEINDEX	Invalid index for current value of a parameter.
PEP_ERR_LINKEDPARAMVALUE	Operation illegal on linked parameter.
PEP_ERR_INVALIDPARAMETERRANGEINFO	Invalid PEP_RI_XXX combination
PEP_ERR_INVALIDSORTMETHODINDEX	Invalid sort method index.

Constant	*Description*
PEP_ERR_INVALIDGRAPHSUBTYPE	Invalid PEP_GST_XXX or PEP_GST_XXX does not match PEP_GT_XXX or PEP_GST_XXX current graph type.
PEP_ERR_BADGRAPHOPTIONINFO	One of the members of PEGraphOptionInfo is out of range.
PEP_ERR_BADGRAPHAXISINFO	One of them members of PEGraphAxisInfo is out of range.
PEP_ERR_NOTEXTERNALSUBREPORT	The subreport is not imported.
PEP_ERR_INVALIDPARAMETERVALUE	
PEP_ERR_INVALIDFORMULASYNTAXTYPE	Specified formula syntax not in PEP_FS_XXX.
PEP_ERR_INVALIDCROPVALUE	
PEP_ERR_INVALIDCOLLATIONVALUE	
PEP_ERR_STARTPAGEGREATERSTOPPAGE	
PEP_ERR_OTHERERROR	
PEP_ERR_INTERNALERROR	Programming error.
PEP_ERR_NOTIMPLEMENTED	

Event Id Constants

Constant	*Description*
PEP_CLOSE_PRINT_WINDOW_EVENT	
PEP_ACTIVATE_PRINT_WINDOW_EVENT	
PEP_DEACTIVATE_PRINT_WINDOW_EVENT	
PEP_PRINT_BUTTON_CLICKED_EVENT	
PEP_EXPORT_BUTTON_CLICKED_EVENT	
PEP_ZOOM_LEVEL_CHANGING_EVENT	
PEP_FIRST_PAGE_BUTTON_CLICKED_EVENT	
PEP_PREVIOUS_PAGE_BUTTON_CLICKED_EVENT	
PEP_NEXT_PAGE_BUTTON_CLICKED_EVENT	
PEP_LAST_PAGE_BUTTON_CLICKED_EVENT	
PEP_CANCEL_BUTTON_CLICKED_EVENT	
PEP_CLOSE_BUTTON_CLICKED_EVENT	
PEP_SEARCH_BUTTON_CLICKED_EVENT	
PEP_GROUP_TREE_BUTTON_CLICKED_EVENT	
PEP_PRINT_SETUP_BUTTON_CLICKED_EVENT	
PEP_REFRESH_BUTTON_CLICKED_EVENT	

Constant	*Description*
PEP_SHOW_GROUP_EVENT	
PEP_DRILL_ON_GROUP_EVENT	Include drill on graph.
PEP_DRILL_ON_DETAIL_EVENT	
PEP_READING_RECORDS_EVENT	
PEP_START_EVENT	
PEP_STOP_EVENT	
PEP_MAPPING_FIELD_EVENT	
PEP_RIGHT_CLICK_EVENT	Right mouse click.
PEP_LEFT_CLICK_EVENT	Left mouse click.
PEP_MIDDLE_CLICK_EVENT	Middle mouse click.
PEP_DRILL_ON_HYPERLINK_EVENT	
PEP_LAUNCH_SEAGATE_ANALYSIS_EVENT	

Formula Syntax Constants

Constant	*Value*
PEP_FST_CRYSTAL	0
PEP_FST_BASIC	1

Graph Options Constants

The following Graph (Chart) Options Constants are listed in this section.

Graph Axis Division Method Constants

Constant	*Value*
PEP_ADM_AUTOMATIC	0
PEP_ADM_MANUAL	1

Graph Placement Constants

Constant
PEP_GLP_PLACEUPPERRIGHT
PEP_GLP_PLACEBOTTOMCENTER
PEP_GLP_PLACETOPCENTER
PEP_GLP_PLACERIGHT
PEP_GLP_PLACELEFT

Graph Gridline Constants

Constant
PEP_GGT_NOGRIDLINES
PEP_GGT_MINORGRIDLINES
PEP_GGT_MAJORGRIDLINES
PEP_GGT_MAJORANDMINORGRIDLINES

Graph Bar Size Constants

Constant
PEP_GBS_MINIMUMBARSIZE
PEP_GBS_SMALLBARSIZE
PEP_GBS_AVERAGEBARSIZE
PEP_GBS_LARGEBARSIZE
PEP_GBS_MAXIMUMBARSIZE

Graph Pie Size Constants

Constant
PEP_GPS_MINIMUMPIESIZE
PEP_GPS_SMALLPIESIZE
PEP_GPS_AVERAGEPIESIZE
PEP_GPS_LARGEPIESIZE
PEP_GPS_MAXIMUMPIESIZE

Graph Detached Pie Slice Constants

Constant
PEP_GDPS_NODETACHMENT
PEP_GDPS_SMALLESTSLICE
PEP_GDPS_LARGESTSLICE

Graph Legend Layout Constants

Constant	*Value*	*Description*
PEP_GLL_PERCENTAGE	0	
PEP_GLL_AMOUNT	1	
PEP_GLL_CUSTOM	2	Use for PEGetGraphOptionInfo. Do not use in PESetGraphOptionInfo.

Graph Marker Size Constants

Constant
PEP_GMS_SMALLMARKERS
PEP_GMS_MEDIUMSMALLMARKERS
PEP_GMS_MEDIUMMARKERS
PEP_GMS_MEDIUMLARGEMARKERS
PEP_GMS_LARGEMARKERS

Graph Marker Shape Constants

Constant
PEP_GMSP_RECTANGLESHAPE
PEP_GMSP_CIRCLESHAPE
PEP_GMSP_DIAMONDSHAPE
PEP_GMSP_TRIANGLESHAPE

Graph Color Constants

Constant
PEP_GCR_COLORCHART
PEP_GCR_BLACKANDWHITECHART

Chart Data Point Constants

Constant
PEP_GDP_NONE
PEP_GDP_SHOWLABEL
PEP_GDP_SHOWVALUE

Chart Number Format Constants

Constant
PEP_GNF_NODECIMAL
PEP_GNF_ONEDECIMAL
PEP_GNF_TWODECIMAL
PEP_GNF_CURRENCYNODECIMAL
PEP_GNF_CURRENCYTWODECIMAL
PEP_GNF_PERCENTNODECIMAL
PEP_GNF_PERCENTONEDECIMAL
PEP_GNF_PERCENTTWODECIMAL

Graph Viewing Angle Constants

Constant
PEP_GVA_STANDARDVIEW
PEP_GVA_TALLVIEW
PEP_GVA_TOPVIEW
PEP_GVA_DISTORTEDVIEW
PEP_GVA_SHORTVIEW
PEP_GVA_GROUPEYEVIEW
PEP_GVA_GROUPEMPHASISVIEW
PEP_GVA_FEWSERIESVIEW
PEP_GVA_FEWGROUPSVIEW
PEP_GVA_DISTORTEDSTDVIEW
PEP_GVA_THICKGROUPSVIEW
PEP_GVA_SHORTERVIEW
PEP_GVA_THICKSERIESVIEW
PEP_GVA_THICKSTDVIEW
PEP_GVA_BIRDSEYEVIEW
PEP_GVA_MAXVIEW

Graph Title Type Constants

Constant
PEP_GTT_TITLE
PEP_GTT_SUBTITLE
PEP_GTT_FOOTNOTE
PEP_GTT_SERIESTITLE
PEP_GTT_GROUPSTITLE
PEP_GTT_XAXISTITLE
PEP_GTT_YAXISTITLE
PEP_GTT_ZAXISTITLE

Graph Text Font Constants

Constant
PEP_GTF_TITLEFONT
PEP_GTF_SUBTITLEFONT
PEP_GTF_FOOTNOTEFONT
PEP_GTF_GROUPSTITLEFONT
PEP_GTF_DATATITLEFONT
PEP_GTF_LEGENDFONT
PEP_GTF_GROUPLABELSFONT
PEP_GTF_DATALABELSFONT

Graph Type and Subtype Constants

The following Graph Type and Subtype Constants are listed below.

Graph Major Type Constants

Constant	*Description*
PEP_GT_BARCHART	
PEP_GT_LINECHART	
PEP_GT_AREACHART	

Constant	*Description*
PEP_GT_PIECHART	
PEP_GT_DOUGHNUTCHART	
PEP_GT_THREEDRISERCHART	
PEP_GT_THREEDSURFACECHART	
PEP_GT_SCATTERCHART	
PEP_GT_RADARCHART	
PEP_GT_BUBBLECHART	
PEP_GT_STOCKCHART	
PEP_GT_USERDEFINEDCHART	For PEGetGraphTypeInfo only.
PEP_GT_UNKNOWNTYPECHART	Do not use in PESetGraphTypeInfo.

Bar Chart Subtype Constants

Constant
PEP_GST_SIDEBYSIDEBARCHART
PEP_GST_STACKEDBARCHART
PEP_GST_PERCENTBARCHART
PEP_GST_FAKED3DSIDEBYSIDEBARCHART
PEP_GST_FAKED3DSTACKEDBARCHART
PEP_GST_FAKED3DPERCENTBARCHART

Line Chart Subtype Constants

Constant
PEP_GST_REGULARLINECHART
PEP_GST_STACKEDLINECHART
PEP_GST_PERCENTAGELINECHART
PEP_GST_LINECHARTWITHMARKERS
PEP_GST_STACKEDLINECHARTWITHMARKERS
PEP_GST_PERCENTAGELINECHARTWITHMARKERS

Area Chart Subtype Constants

Constant
PEP_GST_ABSOLUTEAREACHART
PEP_GST_STACKEDAREACHART
PEP_GST_PERCENTAREACHART
PEP_GST_FAKED3DABSOLUTEAREACHART
PEP_GST_FAKED3DSTACKEDAREACHART
PEP_GST_FAKED3DPERCENTAREACHART

Pie Chart Subtype Constants

Constant
PEP_GST_REGULARPIECHART
PEP_GST_FAKED3DREGULARPIECHART
PEP_GST_MULTIPLEPIECHART
PEP_GST_MULTIPLEPROPORTIONALPIECHART

Doughnut Chart Subtype Constants

Constant
PEP_GST_REGULARDOUGHNUTCHART
PEP_GST_MULTIPLEDOUGHNUTCHART
PEP_GST_MULTIPLEPROPORTIONALDOUGHNUTCHART

3D Riser Chart Subtype Constants

Constant
PEP_GST_THREEDREGULARCHART
PEP_GST_THREEDPYRAMIDCHART
PEP_GST_THREEDOCTAGONCHART
PEP_GST_THREEDCUTCORNERSCHART

3D Surface Chart Subtype Constants

Constant
PEP_GST_THREEDSURFACEREGULARCHART
PEP_GST_THREEDSURFACEWITHSIDESCHART
PEP_GST_THREEDSURFACEHONEYCOMBCHART

Scatter Chart Subtype Constants

Constant
PEP_GST_XYSCATTERCHART
PEP_GST_XYSCATTERDUALAXISCHART
PEP_GST_XYSCATTERWITHLABELSCHART
PEP_GST_XYSCATTERDUALAXISWITHLABELSCHART

Radar Chart Subtype Constants

Constant
PEP_GST_REGULARRADARCHART
PEP_GST_STACKEDRADARCHART
PEP_GST_RADARDUALAXISCHART

Bubble Chart Subtype Constants

Constant
PEP_GST_REGULARBUBBLECHART
PEP_GST_DUALAXISBUBBLECHART

Stock (High/Low/Close) Chart Subtype Constants

Constant
PEP_GST_HIGHLOWCHART
PEP_GST_HIGHLOWOPENCLOSECHART

Miscellaneous Chart Subtype Constants

Constant
PEP_GST_UNKNOWNSUBTYPECHART

Job Destination Constants

Constant
PEP_TO_NOWHERE
PEP_TO_WINDOW
PEP_TO_PRINTER
PEP_TO_EXPORT
PEP_FROM_QUERY

Job Status Constants

Job Status Constants are returned by *CRPEJob::GetJobStatus, Page 41*, and by *CRPEStopEventInfo, Page 146*, member jobStatus. CRPEJob::GetJobStatus will return -1 if the Seagate Crystal Report Engine has not been opened, or if a print job has not been established.

Constant	*Description*
PEP_JOBNOTSTARTED	The job has not been started.
PEP_JOBINPROGRESS	The job is currently in progress.
PEP_JOBCOMPLETED	The job has been completed successfully.
PEP_JOBFAILED	An error has occurred and the job has failed.
PEP_JOBCANCELLED	The job has been canceled by the user.
PEP_JOBHALTED	The job has too many records or has taken too much time.

Mouse Click Action Constants

Constant
PEP_MOUSE_NOTSUPPORTED
PEP_MOUSE_DOWN
PEP_MOUSE_UP
PEP_MOUSE_DOUBLE_CLICK

Report Prompt Option Constants

Constant	*Value*
PE_RPTOPT_PROMPT_NONE	0
PE_RPTOPT_PROMPT_NORMAL	1
PE_RPTOPT_PROMPT_ALWAYS	2

Section Codes Constants

See *Working with section codes in the Seagate Crystal Reports Technical Reference Guide*.

Constant
PEP_ALLSECTIONS
PEP_SECT_REPORT_HEADER
PEP_SECT_PAGE_HEADER
PEP_SECT_GROUP_HEADER
PEP_SECT_DETAIL
PEP_SECT_GROUP_FOOTER
PEP_SECT_PAGE_FOOTER
PEP_SECT_REPORT_FOOTER

Sort Method Constants

Constant
PEP_OR_NO_SORT
PEP_OR_ALPHANUMERIC_ASCENDING
PEP_OR_ALPHANUMERIC_DESCENDING
PEP_OR_NUMERIC_ASCENDING
PEP_OR_NUMERIC_DESCENDING

Sort Order Constants

Constant	*Description*
PEP_SF_DESCENDING	Sorts data in descending order (Z to A, 9 to 1).
PEP_SF_ASCENDING	Sorts data in ascending order (A to Z, 1 to 9).

Subreport Info Constants

Constant	*Value*
PEP_SRI_ONOPENJOB	0
PEP_SRI_ONFUNCTIONCALL	1

Table Difference Constants

Any combination of Table Difference Constants can be returned from *CRPEJob::CheckNthTableDifferences, Page 21*. The constants are used by **CRPETableDifferenceInfo** (page 147).

Constant	*Value*
PEP_TCD_OKAY	0x00000000
PEP_TCD_DATABASENOTFOUND	0x00000001
PEP_TCD_SERVERNOTFOUND	0x00000002
PEP_TCD_SERVERNOTOPENED	0x00000004
PEP_TCD_ALIASCHANGED	0x00000008
PEP_TCD_INDEXESCHANGED	0x00000010
PEP_TCD_DRIVERCHANGED	0x00000020
PEP_TCD_DICTIONARYCHANGED	0x00000040
PEP_TCD_FILETYPECHANGED	0x00000080
PEP_TCD_RECORDSIZECHANGED	0x00000100
PEP_TCD_ACCESSCHANGED	0x00000200
PEP_TCD_PARAMETERSCHANGED	0x00000400
PEP_TCD_LOCATIONCHANGED	0x00000800
PEP_TCD_DATABASEOTHER	0x00001000
PEP_TCD_NUMFIELDSCHANGED	0x00010000
PEP_TCD_FIELDOTHER	0x00020000
PEP_TCD_FIELDNAMECHANGED	0x00040000
PEP_TCD_FIELDDESCCHANGED	0x00080000
PEP_TCD_FIELDTYPECHANGED	0x00100000
PEP_TCD_FIELDSIZECHANGED	0x00200000
PEP_TCD_NATIVEFIELDTYPECHANGED	0x00400000
PEP_TCD_NATIVEFIELDOFFSETCHANGED	0x00800000
PEP_TCD_NATIVEFIELDSIZECHANGED	0x01000000
PEP_TCD_FIELDDECPLACESCHANGED	0x02000000

Zoom Constants

Constant	*Description*
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.

class CRPEngine and class CRPEJob Obsolete Methods

This section describes the class CRPEngine and class CRPEJob obsolete methods. Documentation is provided for support of older applications. Use of these methods for new applications is not recommended.

CRPEJob::GetMinimumSectionHeight

CRPEJob::GetMinimumSectionHeight is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::GetSectionHeight, Page 64*, for all new development.

CRPEJob::GetMinimumSectionHeight was used to retrieve the minimum section height information for selected sections in the specified report.

Syntax

```
BOOL GetMinimumSectionHeight (
                  short sectionCode,
                  short *height );
```

Parameters

sectionCode	Specifies the code for the report section for which you want to retrieve the minimum height. Use one of the PEP_XXX **Section Codes Constants** (page 174).
height	Specifies a pointer to the minimum height, in twips, for the specified report section.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::GetNParams

CRPEJob::GetNParams is now obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::GetNParameterFields, Page 46*, for all new development.

CRPEJob::GetNParams was used to determine how many parameters are required by a stored procedure in an SQL data base table. This method was usually used in conjunction with *CRPEJob::GetNthParam, Page 177*, or *CRPEJob::SetNthParam, Page 179*.

Syntax

```
short GetNParams ();
```

Returns

- Returns the number of parameters in the current stored procedure being used to generate the report.
- Returns -1 if an error occurs.

CRPEJob::GetNthParam

CRPEJob::GetNthParam is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::GetNthParameterField, Page 52*, for all new development.

CRPEJob::GetNthParam was Used to determine a particular parameter required by a stored procedure in a SQL database table.

Syntax

```
BOOL GetNthParam (
           short paramN,
           CString &paramValue );
```

Parameters

paramN	Specifies which parameter in the stored procedure you want to retrieve. The first parameter of a stored procedure is 0, the second is 1, etc.
paramValue	Reference to the CString containing the current value of the specified parameter in the stored procedure.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Remarks

To determine whether a parameter field is a stored procedure, use *CRPEJob::GetNthParameterField, Page 52*.

CRPEJob::GetNthParamInfo

CRPEJob::GetNthParamInfo is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::GetParameterValueInfo, Page 59*, for all new development.

CRPEJob::GetNthParamInfo was used to retrieve the name and type of a specified stored procedure parameter.

Syntax

```
BOOL GetNthParamInfo (
           short paramN,
           CRPEParameterInfo *paramInfo );
```

Parameters

paramN	Specifies the number of the stored procedure parameter about which you want to retrieve information.
paramInfo	Specifies a pointer to an (obsolete) **CRPEParameterInfo** (page 180) structure.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetMinimumSectionHeight

CRPEJob::SetMinimumSectionHeight is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::SetSectionHeight, Page 99*, for all new development.

CRPEJob::SetMinimumSectionHeight was used to set the minimum section height information for selected sections in the specified report.

Syntax

```
BOOL SetMinimumSectionHeight (
          short sectionCode,
          short height );
```

Parameters

sectionCode	Specifies the section for which you want to specify the minimum height. Use one of the PEP_XXX **Section Codes Constants** (page 174).
height	Specifies the minimum height in twips of the specified section.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::SetNthParam

CRPEJob::SetNthParam is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use *CRPEJob::SetNthParameterField, Page 89*, for all new development.

CRPEJob::SetNthParam was used to set the value of a parameter in a stored procedure.

Syntax

```
BOOL SetNthParam (
          short paramN,
          const char *paramValue );
```

Parameters

paramN	Specifies for which parameter in the stored procedure you want to set the value. The first parameter of a stored procedure is 0, the second is 1, etc.
paramValue	Specifies a pointer to the new value of the indicated parameter. This value must be a string. Please see Remarks below.

Returns

- TRUE if the call is successful.
- FALSE 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 (for example: “100”).

class CRPEngine and class CRPEJob Obsolete Structures

This section describes the class CRPEngine and class CRPEJob obsolete structure. Documentation is provided for support of older applications. Use of this structure for new applications is not recommended.

CRPEParameterInfo

CRPEParameterInfo is obsolete. It will still work for older applications. However, there have been a number of major enhancements to the parameter fields (default values, multiple values, value ranges, etc.). Use **CRPEParameterValueInfo** (page 134), for all new development.

This structure was used to contains information about a parameter for a stored procedure in an SQL database. It was used by *CRPEJob::GetNthParamInfo, Page 178*.

Data Members

m_Type	WORD	Specifies the data type of the parameter. Use the appropriate constant from the list below (these data types are supported by The Crystal Report Engine):
	Data Type	***Constant***
	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
m_Name	const _TCHAR	Specifies the name of the parameter in a null-terminated string.

Constructor CRPEParameterInfo::CRPEParameterInfo

This constructs an (obsolete) CRPEParameterInfo structure object. It assigns default values to the members of the structure.

Constructor Syntax

```
CRPEParameterInfo ();
```

2 Crystal ActiveX Control Reference

What you will find in this chapter...

Overview of Section Codes

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.»

ActiveX Controls Properties

The following Properties, listed alphabetically, are discussed in this section.

Properties A...O

Properties P...V

Properties W...Z (Window...)

Action

Action specifies the trigger for the printing of the report.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

BoundReportFooter

BoundReportFooter indicates whether or not a footer is printed at the bottom of each page with a page number when printing a bound report.

Syntax

```
[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 239*, 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

BoundReportHeading indicates whether or not a report title is displayed in the Page Header section at the top of each page of a bound report.

Syntax

```
[form.]Report.BoundReportHeading[= Title$]
```

For example:

```
CrystalReport1.BoundReportHeading = “Box Office Report”
```

«Specifies that the title “Box Office Report” be printed at the top of each page of the report.»

Remarks

- This property is ignored if the *ReportSource, Page 239*, is set to 0.
- If this property is left blank, no report title will be printed.

Data Type

String

Availability

Design Time; Runtime

Connect

Connect specifies the information required to log on to an SQL server or an ODBC data source.

Syntax

```
[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 211*.
- 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

CopiesToPrinter

CopiesToPrinter specifies the number of copies to be printed if you are printing to a printer. *Destination, Page 192*, will specify that the job be sent to a printer rather than a window, etc.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

DataFiles

DataFiles specifies the location of the database files or tables used in the report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

DataSource

DataSource specifies the Data Control from which the bound report will retrieve the data.

Remarks

- This property is ignored if the *ReportSource, Page 239*, 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

Destination specifies the destination to which your report is to be printed (Window, Printer, File or Mail).

Syntax

```
[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 225*, and the *PrintFileType, Page 228*.)
- 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, Page 228*.)
- 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, Page 228*.)
- 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 in the *Seagate Crystal Reports Technical Reference Guide*

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

PEExportTo in the *Seagate Crystal Reports Technical Reference Guide*

DetailCopies

DetailCopies specifies the number of copies of each record in the Details section that the program is to print.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

DialogParentHandle

DialogParentHandle specifies the handle of the parent window. The program uses this handle to determine the window within which it centers any dialog boxes that it displays (progress dialog boxes, parameter field prompt dialog boxes, etc.).

Syntax

```
[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, Page 255*, *WindowTop, Page 266*, *WindowHeight, Page 254*, and *WindowWidth, Page 266* properties.
- This property/method is available for subreports.

Data Type

Long Integer

Availability

Runtime only

Related Report Engine Functions

PESetDialogParentWindow in the *Seagate Crystal Reports Technical Reference Guide*

DiscardSavedData

DiscardSavedData indicates whether or not the data that is saved with a report should be discarded.

Syntax

```
[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:

- Do not discard the data: False = 0
- Discard the data: True = 1

This property/method is available for subreports.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEDiscardSavedData in the *Seagate Crystal Reports Technical Reference Guide*

EMailCCList

EMailCCList specifies the "CC" list to which you want your e-mail message sent.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

PEGetExportOptions in the *Seagate Crystal Reports Technical Reference Guide*

EMailMessage

EMailMessage specifies the string that you want to appear as the body of your e-mail message.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

EMailSubject

EMailSubject specifies the subject line in your e-mail message.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

EMailToList

EMailToList specifies the "To" list to which you want your e-mail message sent.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

EMailVIMBCCList

EMailVIMBCCList specifies the "Blind CC" list to which you want your e-mail message sent.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ExchangeFolder

ExchangeFolder specifies the Exchange path to which a file will be exported, when you want to export to Microsoft Exchange.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ExchangePassword

Specifies the Exchange password when you want to export to Microsoft Exchange.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ExchangeProfile

ExchangeProfile specifies the Exchange Profile when you want to export to Microsoft Exchange.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

Formulas

Formulas specifies a new string for an existing formula.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

GraphData

GraphData specifies the data used for a specified chart.

Syntax

```
[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.»

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.

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 Overview of Section Codes, Page 182.
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

Parameter	*Description*	*Expected value*
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 VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

GraphOptions

GraphOptions specifies several of the options for the specified chart.

Syntax

```
[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) = “GROUPHDR.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.

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 Overview of Section Codes, Page 182.
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.

Note: If you are currently using the VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

GraphText

GraphText specifies the various text components for the specified chart.

Syntax

```
[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 *Overview of Section Codes, Page 182*).
- 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.

- *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 VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

GraphType

GraphType specifies the kind of chart used in the selected section in the specified report.

Syntax

```
[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.»

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.

- Select sectionCode from the section code table, see *Overview of Section Codes, Page 182*.
- 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:

For this type of graph	***Use this code for graphType***
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

Note: If you are currently using the VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

GroupCondition

GroupCondition specifies what kind of change in the Group Condition Field will trigger the creation of a group.

Syntax

```
[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	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
Condition for all other data types	
Any Change	ANYCHANGE

Note: If you are currently using the 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 in the *Seagate Crystal Reports Technical Reference Guide*

GroupSelectionFormula

GroupSelectionFormula specifies the groups to be used when printing the report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

GroupSortFields

GroupSortFields specifies the group field(s) that are to be used to sort your data when the report is printed.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

LastErrorNumber

LastErrorNumber specifies the error code for the last runtime error.

Syntax

```
[form.]Report.LastErrorNumber
```

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 in the *Seagate Crystal Reports Technical Reference Guide*

LastErrorString

LastErrorString specifies the error string for the last runtime error.

Syntax

```
[form.]Report.LastErrorString
```

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 in the *Seagate Crystal Reports Technical Reference Guide*

PEGetErrorText in the *Seagate Crystal Reports Technical Reference Guide*

LogOnInfo

LogOnInfo specifies the information required to log on to one or more SQL servers or password-protected databases.

Syntax

```
[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 *Connect, Page 188*, 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 276*, can be used to populate this property with log on information automatically.

Note: If you are currently using the VBX control in your application, you will not be able to print individual subreports.

Data Type

Array of strings

Availability

Runtime

MarginBottom

MarginBottom specifies the bottom margin for the specified report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

MarginLeft

MarginLeft specifies the left margin for the specified report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

MarginRight

MarginRight specifies the right margin for the specified report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

MarginTop

MarginTop specifies the top margin for the specified report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

ParameterFields

ParameterFields specifies the default value of the specified parameter field. When the prompting dialog box appears for the parameter field, the value specified with this property will be the default value that appears. This property is not available for subreports.

Syntax

```
[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”
    ```

  - To change the value in a Date parameter field use the following syntax (user will not be prompted to enter a value):

    ```
    CrystalReport1.ParameterFields(0) = “DatePar;Date(1998,12,7);TRUE”
    ```

  - 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”
    ```

Data Type

Array of strings

Availability

Runtime only

Related Report Engine Functions

PESetNthParameterField in the *Seagate Crystal Reports Technical Reference Guide*

Password

Password specifies the password needed to use database tables on a restricted Access .MDB file.

Syntax

```
[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.

- In Microsoft Access 95 and later, an Access database can have session security (also know 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 property. If the Access database contains database-level security, use a linefeed character, Chr(10), followed by the database-level password. For example:

```
CrystalReport1.Password = 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:

```
CrystalReport1.Password "sesspswd" & Chr(10) & "dbpassword"
```

Alternately, database-level security can also be handled by assigning the database-level password to the Password parameter of the *LogOnInfo, Page 211*, property.

Related Report Engine Functions

PESetNthTableSessionInfo in the *Seagate Crystal Reports Technical Reference Guide*

PrintDay

PrintDay specifies the day component of the print date (if different from the actual date the report is printed).

Syntax

```
[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 233*, *PrintMonth, Page 233*, 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrinterCollation

PrinterCollation indicates whether or not copies will be collated when printed, if more than one copy is to be printed. Use *PrinterCopies, Page 218*, to specify how many copies should be printed.

Syntax

```
[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 not available for subreports.

Note: If you are currently using the 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrinterCopies

PrinterCopies specifies the number of report copies to be printed. *PrinterCollation, Page 218*, indicates whether or not the copies will be collated when printed.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrinterDriver

PrinterDriver specifies the name of the printer driver that is to be used when the report is printed.

Syntax

```
[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 220*, and *PrinterPort, Page 221*, 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 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

PrinterName

PrinterName specifies the name of the printer that is to print the report.

Syntax

```
[form.]Report.PrinterName[= PrinterName$]
```

For example:

```
CrystalReport1.PrinterName = "Epson LQ-850"
```

«Specifies the Epson LQ-850 printer.»

Remarks

- *PrinterDriver, Page 219*, PrinterName, and *PrinterPort, Page 221*, 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 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

PrinterPort

PrinterPort specifies the name of the printer port for the specified printer.

Syntax

```
[form.]Report.PrinterPort[= PortName$]
```

For example:

```
CrystalReport1.PrinterPort= "LPT1"
```

«Sets the printer port to LPT1.»

Remarks

- *PrinterDriver, Page 219*, *PrinterName, Page 220*, 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 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

PrinterStartPage

PrinterStartPage specifies the first page to be printed.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrinterStopPage

PrinterStopPage specifies the last page to be printed.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileCharSepQuote

PrintFileCharSepQuote specifies the quote character used to enclose alphanumeric field data when printing to a file using character-separated format.

Syntax

```
[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 228*, is 5 - Character-separated values.
- Applies only when *Destination, Page 192*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileCharSepSeparator

PrintFileCharSepSeparator specifies the character(s) you want to use to separate the fields when printing to a file using the Character Separated Value format.

Syntax

```
[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 228*, is 5 - Character-separated values.
- Applies only when *Destination, Page 192*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileLinesPerPage

PrintFileLinesPerPage specifies the number of lines to be printed before the page break. The default is 60 lines per page.

Syntax

```
[form.]Report.PrintFileLinesPerPage
```

For example:

```
CrystalReport1.PrintFileLinesPerPage = 50
```

«Fifty lines will be printed before a page break.»

Note: If you are currently using the 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

PrintFileName specifies the name of the file to which the report is to be printed.

Syntax

```
[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 192*, 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileODBCPassword

PrintFileODBCPassword specifies the password that you need to connect to the data source, whenever you want to export in ODBC format.

Syntax

```
[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

PrintFileODBCSource specifies the name of the data source to which you want to export, whenever you export in ODBC format.

Syntax

```
[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

PrintFileODBCTable specifies the name of the table in the data source to which you want to export, whenever you want to export in ODBC format.

Syntax

```
[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

PrintFileODBCUser specifies the User ID that you need to connect to the data source, whenever you export in ODBC format.

Syntax

```
[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

PrintFileType specifies the type of print file to use when printing a report to a file.

Syntax

```
[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 192*, 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 224*.

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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileUseRptDateFmt

PrintFileUseRptDateFmt specifies, when printing to a file, 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.

Syntax

```
[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 228*, is 0 - Record, 1 - Tab-separated, 3 - Data Interchange Format (DIF), 4 - CSV, or 5 - Character-Separated.
- Applies only when *Destination, Page 192*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintFileUseRptNumberFmt

PrintFileUseRptNumberFmt specifies, when printing to a file, 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.

Syntax

```
[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 228*, is 0 - Record, 1 - Tab-separated, 3 - Data Interchange Format (DIF), 4 - CSV, or 5 - Character-Separated.
- Applies only when *Destination, Page 192*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintMonth

PrintMonth specifies the month component of the print date (if different from the actual date the report is printed).

Syntax

```
[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 233*, *PrintMonth, Page 233*, and *PrintDay, Page 217*, 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

PrintYear

PrintYear specifies the year component of the print date (if different from the actual date the report is printed).

Syntax

```
[form.]Report.PrintYear[=Year%]
```

For example:

```
CrystalReport1.PrintYear = 1999
```

«Sets the year component of the print date to 1999.»

Remarks

- Enter the print year as a four-digit number.
- *PrintYear, Page 233*, *PrintMonth, Page 233*, and *PrintDay, Page 217*, 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

ProgressDialog

ProgressDialog indicates whether the display of the Progress dialog box is enabled or disabled. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

Syntax

```
[form.]Report.ProgressDialog[= {True|False}]
```

For example:

```
CrystalReport1.ProgressDialog = 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 in the *Seagate Crystal Reports Technical Reference Guide*

RecordsPrinted

RecordsPrinted specifies the number of records actually printed.

Syntax

```
[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 236*.
- This property/method is available for subreports.

Data Type

Long

Availability

Runtime

Related Report Engine Functions

PEGetJobStatus in the *Seagate Crystal Reports Technical Reference Guide*

RecordsRead

RecordsRead specifies the number of records actually processed.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

RecordsSelected

RecordsSelected specifies the number of records selected for inclusion in the report out of the total number of records read.

Syntax

```
[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 235*. 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 in the *Seagate Crystal Reports Technical Reference Guide*

ReportDisplayPage

ReportDisplayPage indicates which page of a multi-page report is currently being displayed in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ReportFileName

ReportFileName specifies the name of the report to be printed.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ReportLatestPage

ReportLatestPage specifies the last page printed in the specified report.

Note: ReportLatestPage will only contain the last page number after calling CrystalReport1.PageCount.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ReportSource

ReportSource specifies the source of the report as a report file, a Visual Basic data control, or a True Grid data control.

Syntax

```
[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

ReportStartPage specifies the first page printed in the specified report.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

ReportTitle

ReportTitle specifies a title for the report.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

SectionFont

SectionFont specifies the font for one or more sections in the specified report.

Syntax

```
[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 *Overview of Section Codes, Page 182*.
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*
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 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 in the *Seagate Crystal Reports Technical Reference Guide*

SectionFormat

SectionFormat specifies the format for one or more sections in the specified report.

Syntax

```
[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;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;X;255.0.0”
```

when making changes to the DETAIL section only, but

```
CrystalReport1.SectionFormat(0) = “HEADER;T;F;F;X;X;X;X;X;255.0.0”
CrystalReport1.SectionFormat(1) = “DETAIL;T;F;F;X;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:

sectionCode	Please refer to the *Overview of Section Codes, Page 182*.
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>.<B> 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 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 in the *Seagate Crystal Reports Technical Reference Guide*

SectionLineHeight

SectionLineHeight specifies the line height in twips.

Syntax

```
[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

A twip is 1/1440 inch; there are 20 twips in a point.

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:

sectionCode	Specifies the section code for the report section(s) for which you want to set a new line height. See *Overview of Section Codes, Page 182*.
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.

Note: If you are currently using the VBX control in your application, you will not be able to print individual subreports.

Data Type

Arrays of strings

Availability

Runtime

SectionMinHeight

SectionMinHeight specifies the minimum section height for the specified report section.

Syntax

```
[form.]Report.SectionMinHeight (SequentialIndex%)[=sectionCode;minHeight$]
```

For example:

```
CrystalReport1.SectionMinHeight(0) = “DETAIL;500”
```

«Sets the minimum height for details section 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

PESetNDetailCopies in the *Seagate Crystal Reports Technical Reference Guide*

SelectionFormula

SelectionFormula specifies the records to be used when printing the report.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

SessionHandle

SessionHandle specifies the session handle for a user once the *UserName, Page 251*, and *Password, Page 216*, properties have opened an Access.mdb file for use by the report.

Syntax

```
[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, Page 216* and *UserName, Page 251* 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 VBX control in your application, you will not be able to print individual subreports.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PESetNthTableSessionInfo in the *Seagate Crystal Reports Technical Reference Guide*

SortFields

SortFields specifies the field(s) that are to be used to sort your data when the report is printed.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

SQLQuery

SQLQuery specifies the SQL query string used by the specified report.

Syntax

```
[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 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 in the *Seagate Crystal Reports Technical Reference Guide*

Status

Status indicates the print status for the specified report.

Syntax

```
[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" if PEOpenEngine or PEOpenPrintJob 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 job has been halted (PE_JOBHALETED).

This property/method is available for subreports.

Data Type

Integer

Availability

Runtime

Related Report Engine Functions

PEGetSQLQuery in the *Seagate Crystal Reports Technical Reference Guide*

StoredProcParam

StoredProcParam specifies the stored procedure parameters when using a report based on SQL stored procedures.

Syntax

```
[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 VBX control in your application, you will not be able to print individual subreports.

Note: Under the native driver for MS SQL Server, stored procedure input parameters are limited to 64 characters rather than 255.

Data Type

Arrays of strings

Availability

Runtime

Related Report Engine Functions

PESetNthParameterDefaultValue in the *Seagate Crystal Reports Technical Reference Guide*

SubreportToChange

SubreportToChange 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).

Syntax

```
[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	LogonInfo	SectionFormat
DataFiles	MarginBottom	SectionLineHeight
DetailCopies	MarginLeft	SectionMinHeight
Formulas	MarginRight	SelectionFormula
GraphData	MarginTop	SessionHandle
GraphOptions	Password	SortFields
GraphText	PrintDay	SQLQuery
GraphType	PrintMonth	StoredProcParam

GroupCondition	PrintYear	UserName
GroupSelectionFormula	ReportTitle	
GroupSortFields	SectionFont	

- 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 VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Runtime

UserName

UserName specifies the name given to a user for logging on to a protected Access.mdb file in order to obtain data files needed by the report.

Syntax

```
[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 VBX control in your application, you will not be able to print individual subreports.

Data Type

String

Availability

Runtime

Related Report Engine Functions

PESetNthTableSessionInfo in the *Seagate Crystal Reports Technical Reference Guide*

WindowAllowDrillDown

WindowAllowDrillDown indicates whether or not drill-down on summary values is allowed in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowBorderStyle

WindowBorderStyle specifies the type of border for the preview window.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer (Enumerated)

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowControlBox

WindowControlBox to specify 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.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowControls

WindowControls indicates whether or not the print controls are to appear in the preview window when printing a report to a window.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEShowPrintControls in the *Seagate Crystal Reports Technical Reference Guide*

WindowHeight

WindowHeight specifies the height of the preview window when the report is printed to a window.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowLeft

WindowLeft specifies 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).

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowMaxButton

WindowMaxButton indicates 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.

Syntax

```
[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 192*, is set to 0.
- If you set both WindowMaxButton and *WindowMinButton, Page 256*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowMinButton

WindowMinButton indicates 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.

Syntax

```
[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 192*, is set to 0.
- If you set both *WindowMaxButton, Page 256*, 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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowParentHandle

WindowParentHandle specifies the handle of the parent window if the preview window is to be the child of another window.

Syntax

```
[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

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowCancelBtn

WindowShowCancelBtn indicates whether or not a Cancel button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowCloseBtn

WindowShowCloseBtn indicates whether or not a Close button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowExportBtn

WindowShowExportBtn indicates whether or not an Export button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowGroupTree

WindowShowGroupTree indicates whether or not a Group Tree is displayed in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowNavigationCtls

WindowShowNavigationCtls indicates whether or not the Navigation Controls are available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowPrintBtn

WindowShowPrintBtn indicates whether or not a Print button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowPrintSetupBtn

WindowShowPrintSetupBtn indicates whether or not a Print Setup button is available in the preview window.

Syntax

```
[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 261*), 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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowProgressCtls

WindowShowProgressCtls indicates whether or not controls indicating the progress of a report being generated are displayed in the preview window.

Syntax

```
[form.]Report.WindowShowProgressCtls[= {True|False}]
```

For example:

```
CrystalReport1.WindowShowProgressCtls = 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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowRefreshBtn

WindowShowRefreshBtn indicates whether or not a Refresh button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowSearchBtn

WindowShowSearchBtn indicates whether or not a Search button is available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowShowZoomCtl

WindowShowZoomCtl indicates whether or not Zoom Controls are available in the preview window.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowState

WindowState specifies the state of the preview window (normal, minimized, or maximized) when the report is printed.

Syntax

```
[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 in the *Seagate Crystal Reports Technical Reference Guide*

WindowTitle

WindowTitle specifies the title that you want to appear in the preview window title bar when the report is printed to a window.

Syntax

```
[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 192*, = 0.

Data Type

String

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowTop

WindowTop specifies 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.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

WindowWidth

WindowWidth specifies the width of the preview window in pixels.

Syntax

```
[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 192*, is set to 0.

Data Type

Integer

Availability

Design Time; Runtime

Related Report Engine Functions

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

ActiveX Controls Methods

The following Methods are discussed in this section.

FetchSelectionFormula

FetchSelectionFormula returns the selection formula from the current report.

Syntax

```
[form.]Report.FetchSelectionFormula
```

For example:

```
SelectionFormula$= CrystalReport1.FetchSelectionFormula
```

«Retrieves the selection formula from CrystalReport1.»

Remarks

This method does not populate *SelectionFormula, Page 245*, 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

GetNSubreports looks at the report specified in *ReportFileName, Page 237*, and returns the number of subreports in that report.

Syntax

```
[form.]Report.GetNSubreports
```

For example:

```
Number=CrystalReport1.GetNSubreports
```

«Returns the number of subreports in CrystalReport1.»

Remarks

If you are currently using the VBX control in your application, you will not be able to print individual subreports.

GetNthSubreportName

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.

Syntax

```
[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 VBX control in your application, you will not be able to print individual subreports.

LogoffServer

LogoffServer terminates the specified database connection established earlier with the *LogonServer, Page 270*.

Syntax

```
[form.]Report.LogoffServer (connectionId%, boolean allConnections)
```

For example:

```
CrystalReport1.LogoffServer (1, False)
```

«Terminates database connection 1 and only that connection.»

Parameters

connectionId	Integer value that specifies a specific database connection established earlier with LogonServer, Page 270. 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. • True = Terminate all connections. • False = Terminate only the specified connection.

LogonServer

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 269*.

Syntax

```
[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

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: • 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 PEGetNthTableLogOnInfo in the *Seagate Crystal Reports Technical Reference Guide*, the password parameter is undefined.

PageCount

PageCount returns the number of pages in the report.

Syntax

```
[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 187*, to get a valid page count.

PageFirst

PageFirst displays the first page of the report in the preview window.

Syntax

```
[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 187*.

PageLast

PageLast displays the last page of the report in the preview window.

Syntax

```
[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 187*.

PageNext

PageNext displays the next page of the report in the preview window.

Syntax

```
[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 187*.

PagePrevious

PagePrevious displays the previous page of the report in the preview window.

Syntax

```
[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 187*.

PageShow

PageShow displays a specific page of the report in the preview window.

Syntax

```
[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 187*.

PageZoom

PageZoom sets the magnification factor for the report in the preview window to a value from 25% to 400% of the actual size.

Syntax

```
[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 187*.

PageZoomNext

PageZoomNext zooms the magnification level of the report in a preview window to the next default zoom level.

Syntax

```
[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 187*.
- 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

PrinterSelect displays the Printer Selection common dialog box which enables the user to specify a different printer.

Syntax

```
[form.]Report.PrinterSelect
```

For example:

```
CrystalReport1.PrinterSelect
```

«When the user prints, a dialog box appears for selecting the desired printed 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

PrintReport triggers the printing of the report.

Syntax

```
[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 187*. 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 VBX control in your application, you will not be able to print individual subreports.

Availability

Runtime

Related Report Engine Functions

PEStartPrintJob in the *Seagate Crystal Reports Technical Reference Guide*

ReplaceSelectionFormula

ReplaceSelectionFormula overrides the selection formula from the current report with the string that is passed.

Syntax

```
[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 245*, and DOES conflict with setting the property. You cannot set the *SelectionFormula, Page 245*, 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

Reset resets the value of all properties (except *DataSource, Page 191*) to their default values.

Syntax

```
[form.]Report.Reset
```

For example:

```
CrystalReport1.Reset
```

«Returns all properties (except DataSource) for CrystalReport1 to their default values.»

RetrieveDataFiles

RetrieveDatafiles retrieves all "table" locations from the current report, populates *DataFiles, Page 190*, and returns the number of "tables" in the report.

Syntax

```
[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 237*, has been set.

Availability

Runtime only

RetrieveLogonInfo

RetrieveLogonInfo retrieves logon information (except for the password) for all "tables" in the current report, populates *LogOnInfo, Page 211*, and returns the number of "tables" in the report.

Syntax

```
[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 237*, has been set. This method DOES NOT use the string in the Connect property and DOES conflict with setting the property. You cannot 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

RetrieveSQLQuery retrieves the SQL Query from the current report and populates *SQLQuery, Page 247*.

Syntax

```
[form.]Report.RetrieveSQLQuery
```

For example:

```
CrystalReport1.RetrieveSQLQuery
```

«Retrieves the SQL query from CrystalReport1.»

Remarks

This method can only be called AFTER *ReportFileName, Page 237*, has been set.

Availability

Runtime only

RetrieveStoredProcParams

RetrieveStoredProcParams retrieves all stored procedure parameters from the current report, populates *StoredProcParam, Page 249*, and returns the number of parameters.

Syntax

```
[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 237*, has been set.

Availability

Runtime only

SpecifyDataSourceField

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.

Syntax

```
[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.

ACTIVEX CONTROLS ERROR MESSAGES

Error #	*Error*	*Message/Explanation*
400	PE_ERR_CONFLICT LOGON INFOCONNECT	**"Do not use both Connect property and LogonInfo property."** This message appears if you try to set both of these properties at the same time.
401	PE_ERR_REPORTFILE NAMENOTSET	**"ReportFileName property must be set before calling method."** 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: • FetchSelectionFormula • ReplaceSelectionFormula • RetrieveDataFiles • RetrieveSQLQuery • RetrieveStoredProcParams • GetNSubreports
402	PE_ERR_SELECTION FORMULACONFLICT	**"Do not use both SelectionFormula and ReplaceSelectionFormula."** ReplaceSelectionFormula is a method. This message appears if you attempt to use it after SelectionFormula.
403	PE_ERR_BADDLL	**"DLL name must be specified."** This message appears if you call the LogOnServer method and don't specify the DLLname parameter.
404	PE_ERR_STRING TOOLONG	**"A string is too long."** This message appears: • 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.
405	PE_ERR_BAD SUBREPORTINDEX	**"Invalid Subreport index."** This message appears if you call the method GetNthSubreportName and the first parameter does not correspond to a subreport.

Error #	*Error*	*Message/Explanation*
406	PE_ERR_BAD SUBREPORTNAME	**"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_ODBCTABLE NOTSET	**"Need ODBC table name for export."** This message appears when you attempt to export to ODBC without setting the PrintFileODBCTable property.

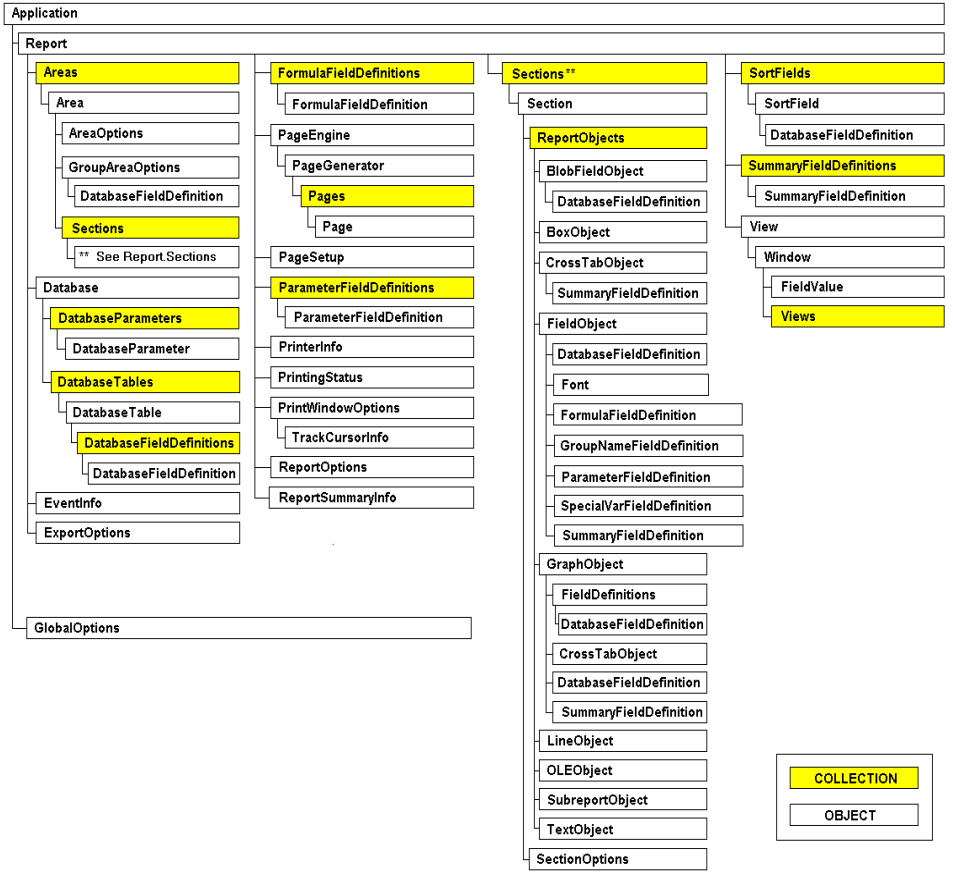
3 Crystal Report Engine Object Model for the Automation Server

What you will find in this chapter...

Overview of the Crystal Report Engine Object Model

Object Hierarchy

The Crystal Report Engine Object Model for the Crystal Report Engine Automation Server provides all of the power of the Crystal Report Engine in an easy to use object-oriented hierarchy. The following diagram illustrates this object hierarchy:



Object Naming Conflicts

If your project includes other libraries that contain objects named identically to those found in the Crystal Report Engine Object Model, you will encounter conflicts unless you reference the objects by using a prefix.

For example, if you have included the DAO Library with the Crystal Report Engine Object Model 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 Model, 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 Model Events

Events are only available when using Visual Basic 5.0 or later. 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 CRPEAuto.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* in the *Seagate Crystal Reports Technical Reference Guide*.

CRYSTAL REPORT ENGINE OBJECT MODEL REFERENCE

The rest of this chapter lists all objects in the Report Engine Object Model and describes their properties, methods, and events. Objects are listed in alphabetical order with a brief description and example of their use in Visual Basic code. A table follows each object description, listing all properties for the object, the purpose of each property, and whether the property is read only or both read and write. After each property table is a listing of object methods in alphabetical order. Finally, if an object supports events, the events are also listed in alphabetical order. Each method and event listing includes a complete description of its purpose and use inside Visual Basic. For more information on using the Report Engine Object Model in Visual Basic, see *Seagate Crystal Reports Technical Reference Guide*.

Objects and Collections

...including Properties, Methods, and Events

The following Objects and Collections are discussed in this section.

Objects and Collections are listed alphabetically; Properties, Methods and Events are grouped under the appropriate Object or Collection.

Application Object

The Application Object is the only creatable object in the Crystal Report Engine Object Model. 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 New keyword or using the CreateObject function and the Prog Id Crystal.CRPE.Application. Note that when using the New keyword, "Crystal.CRPE.Application" does not have to be referenced.

For example,

- Using the New keyword:

```
Dim app as New Application
Set app = New Application
```

Or,

```
’ Automatically creates a new instance of the object when it is first
’ referenced in the code, so it doesn’t have to be set.
Dim app As New Application
```

- Using the CreateObject function:

```
Dim app As Application
Set app = CreateObject(“Crystal.CRPE.Application”)
```

Application Object Properties

Property	*Description*	*Read/Write*
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 **GlobalOptions Object** (page 320), 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.

Application Object Methods

CanClose

The CanClose method indicates whether or not the **Application Object** (page 289), can be destroyed. This method will return FALSE as long as there are valid Report objects in existence. The Application object can be destroyed only if no instances of the **Report Object** (page 347), 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 by using **LogOnServer** (page 291).

Syntax

```
object.LogOffServer DLLName, ServerName, DatabaseName, UserID, Password
```

Parameters

DLLName	Specifies the name of the DLL for the server or password-protected non-SQL table that 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 logon 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 is already set in the report. If you want to override a value that is already set in the report, use a non-empty string (for example, "Server A").

Remarks

If you try to log off a server that is still in use (that is, 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 347*, 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 by using this method, you will remain logged on until you call *LogOffServer, Page 291*, or until the *Application Object, Page 289*, is destroyed. This method corresponds to *PELogOnServer (Seagate Crystal Reports Technical Reference Guide)*, of the Crystal Report Engine API.

Syntax

```
object.LogOnServer DLLName, ServerName, DatabaseName, UserID, Password
```

Parameters

DLLName	Specifies the name of the DLL for the server or password-protected non-SQL table that 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 logon 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 is already set in the report. If you want to override a value that is already set in the report, use a non-empty string (for example, "Server A").

OpenReport

The OpenReport method opens an existing report file, creating an instance of the Report object. Through the *Report Object, Page 347*, you can change formatting, formulas, selection formulas, and sort fields for the report, then print, preview, or export the report. This method corresponds to *PEOpenPrintJob (Seagate Crystal Reports Technical Reference Guide)*, of the Crystal Report Engine API.

Syntax

```
Dim rep As Report
Set rep = app.OpenReport(ReportFilePath)
```

Parameters

ReportFilePath	Specifies a string value indicating the file name and path of the report that you want to open.

Returns

- Returns an instance of the *Report Object, Page 347*, 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 (for example, 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.

Area Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that 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
GroupOptions	Returns **GroupAreaOptions Object** (page 324), 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 (that is, Details, Report Header, Page Footer, etc.).	Read only
	Constant	***Value***
	crDetail	4
	crGroupFooter	5
	crGroupHeader	3
	crPageFooter	7
	crPageHeader	2
	crReportFooter	8
	crReportHeader	1
Property	***Description***	***Read/Write***
Options	Returns **AreaOptions Object** (page 294), which provides properties for getting and setting area options (that is, new page before, keep together, etc.).	Read only

Parent	Reference to the Parent object (**Area Object** (page 293)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
Sections	Returns **Sections Collection** (page 361), 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 (that is, new page before, keep together, underlay section, etc.). An AreaOptions object is obtained from the Options property of the *Area Object, Page 293*.

Each report area is comprised of sections which can be formatted independently of each other by using the *SectionOptions Object, Page 359*. These section options are combined with the options set by using the AreaOptions Object to make up the final appearance of a section and an 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 to which they are set.

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.

AreaOptions Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that 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
NewPageBefore Formula	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 (*Area Object, Page 293*).	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 **Report Object** (page 347).	Read only
ResetPageNumberAfter	Returns/Sets Boolean value indicating if the page number is reset to one after this area is printed.	Read/Write
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 293*, in the collection using the Item property.

Areas Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of areas in the collection.	Read only

Property	*Description*	*Read/Write*
Item	Returns **Area Object** (page 293). Item has an index parameter that can be either a string reference to the area (that is, "RH", "PH", "GHn", "D", "GFn", "PF", or "RF") or a numeric, 1-based index (that is, Item (1)) for the Report Header area. The items in the collection are indexed in the order that they are listed for each section/area.	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Remarks

Instead of using the Item property as shown, you can reference an area directly (for example, Areas ("RH") or Areas (1)).

BlobFieldObject Object

The BlobFieldObject Object allows you to get and set information for bitmap database fields in a report.

BlobFieldObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Field	Returns **DatabaseFieldDefinition Object** (page 300), containing information about the BLOB field.	Read only
Kind	Returns CRObjectKind object which specifies what "kind" of object (that is, box, cross-tab, field, etc.).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2

Property	*Description*	*Read/Write*
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

BoxObject Object

The BoxObject object represents a box that has been drawn on the report. This object allows you to get information about boxes in a report.

BoxObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Kind	Returns CRObjectKind object, which specifies what "kind" of object (that is, box, cross-tab, field, etc.).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Property	*Description*	*Read/Write*
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

CrossTabObject Object

The CrossTabObject Object allows you to get and set information for cross-tab objects in a report.

CrossTabObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Kind	Returns CRObjectKind object, which specifies what "kind" of object (that is, box, cross-tab, field, etc.).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Property	***Description***	***Read/Write***
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
SummaryFields	Returns the **SummaryFieldDefinitions Collection** (page 369), 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 281*.

Database Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Parameters	Returns **DatabaseParameters Collection** (page 303), which specifies the database parameters used in the report.	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
Tables	Returns **DatabaseTables Collection** (page 308), which specifies the database objects used in the report (that is, an Access report may contain a query or an 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

Database Object Methods

Verify, Page 299

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.

Database FieldDefinition Object Properties

Property	*Definition*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
DatabaseFieldName	Specifies the name of the field in the database (that is, Product ID).	Read only
Kind	Returns CRFieldKind, which specifies what "kind" of field (that is, database, summary, formula, etc.).	Read only
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5
	crParameterField	6
	crSpecialVarField	4
	crSummaryField	3
Property	***Definition***	***Read/Write***
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 (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	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
	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
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 300*, in the collection by using the Item property.

Database FieldDefinitions Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	The number of **DatabaseFieldDefinition Object** (page 300), in the collection.	Read only
Parent	Reference to the Parent object (**DatabaseTable Object** (page 304)).	Read only

Property	*Description*	*Read/Write*
Report	Reference to **Report Object** (page 347).	Read only
Item	Returns **DatabaseFieldDefinition Object** (page 300). Item has an index parameter that can be either a string reference to the database field name or a numeric, 1-based index (that is, 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 that 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 DatabaseFieldDefinition(1)).

DatabaseParameter Object

The DatabaseParameter Object specifies a stored procedure in an SQL database or an Access parameterized query.

Database Parameter Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (**Database Object** (page 299)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
Type	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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
	crOleField	20
	crPersistentMemoField	14
	crPictureField	19
	crStringField	12
	crTimeField	11
	crTransientMemoField	13
	crUnknownField	22

Property	***Description***	***Read/Write***
Value	Returns/Sets the value for a database parameter as a string (that is, when you are setting a numeric parameter that 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 302*, in the collection by using the Item property.

DatabaseParameters Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Specifies the number of database parameters in the collection.	Read only

Property	*Description*	*Read/Write*
Item	Returns **DatabaseParameter Object** (page 302). Item has an index parameter that is a numeric, 1-based index (that is, 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 (**Database Object** (page 299)).	Read only
Report	Reference to **Report Object** (page 347).	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.

DatabaseTable Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (that is, PDBDAO.DLL, PDSODBC.DLL, etc.).	Read only
Fields	Returns **DatabaseFieldDefinitions Collection** (page 301), specifying a collection of database fields in the table.	Read only
Location	Gets/sets the location of the database table.	Read/Write
LogOnDatabase Name	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 (for example, 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
Name	Returns/Sets the alias name for the database table used in the report.	Read/Write
Parent	Reference to the Parent object (**Database Object** (page 299)).	Read only

Property	*Description*	*Read/Write*
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 by using the crystal active data driver (see *Active Data Driver* in the *Seagate Crystal Reports Technical Reference Guide*), 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 **GetPrivateData** (page 305), and **SetPrivateData** (page 306), methods.	Read only
Report	Reference to **Report Object** (page 347).	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 **SetSessionInfo** (page 307), method if an Access session has been opened.	Read only
Type	Returns value of type CRDatabaseType (see table below) indicating the type of database (that is, standard or SQL).	Read only
	Constant	***Value***
	crSQLDatabase	2
	crStandardDatabase	1

DatabaseTable Object 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 by using the crystal active data driver, see *Active Data Driver* in the *Seagate Crystal Reports Technical Reference Guide*. GetPrivateData can be used to obtain the active data object assigned to the report by using *SetPrivateData, Page 306*.

Syntax

```
object.GetPrivateData VariantType
```

Parameters

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

DllName	Specifies the DLL used to access the database by the Crystal Report Engine (that is, PDSODBC.DLL).
ServerName	Specifies the name of the server or ODBC data source where the database is located (that is, 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 by using the crystal active data driver (see *Active Data Driver* in the *Seagate Crystal Reports Technical Reference Guide*), 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 *Active Data Driver* in the *Seagate Crystal Reports Technical Reference Guide*.

Syntax

```
object.SetPrivateData DataTag, Data
```

Parameters

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

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 know 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 306*, 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 if the database session, logon, and location information are all correct.
- FALSE if something is wrong.

DatabaseTables Collection

The DatabaseTables Collection is a collection of DatabaseTable objects. A DatabaseTable object exists for every database object (that is, table, query, stored procedure, etc.) accessed by the report. Access a specific *DatabaseTable Object, Page 304*, in the collection by using the Item property.

DatabaseTables Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of database objects in the collection.	Read only
Item	Returns **DatabaseTable Object** (page 304). Item has an index parameter that can be either a string reference to the table, (that is, Item("Catgories")) or a numeric, 1-based index (that is, Item (1)). When a numeric index is used, the items in the collection are indexed in the order that they were added to the report.	Read only
Parent	Reference to the Parent object (**Database Object** (page 299)).	Read only
Report	Reference to **Report Object** (page 347).	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

EventInfo Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
ActivatePrintWindow Event-Enabled	Returns/Sets a Boolean value indicating whether the **ActivatePrintWindow** (page 378), or **DeactivatePrintWindow** (page 380), events of the **Window Object** (page 377), are enabled.	Read/Write
ClosePrintWindow Event-Enabled	Returns/Sets a Boolean value indicating whether the **ClosePrintWindow** (page 379), event of the **Window Object** (page 377), is enabled.	Read/Write
GroupEventEnabled	Returns Boolean value indicating whether the **GroupTreeButtonClicked** (page 383), **ShowGroup** (page 386), or **DrillOnGroup** (page 381), events of the **Window Object** (page 377), are enabled.	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
PrintWindowButton Event-Enabled	Returns Boolean value indicating if events corresponding to preview window buttons for the **Window Object** (page 377),are enabled.	Read only
ReadingRecordsEvent Enabled	Returns Boolean value indicating if the **ReadingRecords** (page 353), event of the **Report Object** (page 347), is enabled.	Read only
Report	Reference to **Report Object** (page 347).	Read only
StartStopEventEnabled	Returns Boolean value indicating if the **Start** (page 354), and **Stop** (page 354), events of the **Report Object** (page 347), are enabled.	Read only

ExportOptions Object

The ExportOptions Object provides properties and methods for retrieving information and setting options for exporting your report (that is, export format, destination, etc.). An ExportOptions Object is obtained from the ExportOptions property of the *Report Object, Page 347*.

ExportOptions Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (that is, disk, mail, etc.).	Read/Write
	Constant	***Value***
	crEDTDiskFile	1
	crEDTMailMAPI	2
	crEDTMailVIM	3
	crEDTMicrosoftExchange	4
	crEDTNoDestination	0
Property	***Description***	***Read/Write***
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
	Constant	***Value***
	crExchangePostDocMessage	1011
	crExchangeFolderType	0

Property	*Description*	*Read/Write*
ExchangeFolderPath	Returns/Sets the path of the Exchange folder for reports exported to Exchange (that is, "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
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 (that is, text, Excel, etc.).	Read/Write
	Constant	***Value***
	crEFTCharSeparatedValues	7
	crEFTCommaSeparatedValues	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
	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
NumberOfLinesPer Page	Returns/Sets the number of lines to appear per page of the report for report formats that are paginated (for example, HTM)L.	Read/Write
ODBCDataSource Name	Returns/Sets the ODBC data source for reports exported to ODBC.	Read/Write
ODBCDataSourcePass word	Returns/Sets the password used to access an ODBC data source for reports exported to ODBC.	Read/Write
ODBCDataSource UserID	Returns/Sets the user name used to access an ODBC data source for reports exported to ODBC.	Read/Write
ODBCExportTable Name	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
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	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

ExportOptions Object Methods

PromptForExportOptions, Page 313

Reset, Page 313

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 FieldDefinitons Collection is a collection of field definitions of all types. This collection is obtained from the DataFields property of the *GraphObject Object, Page 320*. The collection of fields represents the fields used to plot the values on a chart. Access a specific FieldDefinition object in the collection using the Item property.

FieldDefinitions Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of field definitions contained by this collection.	Read only
Item	Returns specific FieldDefnition objects in this collection. Item has an index parameter that is a numeric, 1-based index (that is, Item (1)). The items in the collection are indexed in the order that they are charted.	Read only
Parent	Reference to the Parent object (**GraphObject Object** (page 320)).	Read only
Report	Reference to **Report Object** (page 347).	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 (that is,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 356* (that is, ReportObjects.Item(Index)), where the index references a special field, database field, parameter field, summary field, or group name field (except BLOB field -- see *BlobFieldObject Object, Page 296*).

Field Object Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Field	Returns the field definition object (depending on what the field object is, for example, **DatabaseFieldDefinition Object** (page 300), **FormulaFieldDefinition Object** (page 317), **GroupNameFieldDefinition Object** (page 325), **ParameterFieldDefinition Object** (page 338), **SummaryFieldDefinition Object** (page 367), and **SpecialVarFieldDefinition Object** (page 364)).	Read only
Font	Returns **Font Object** (page 315), with information on what type of font is being used (for example, name).	Read only
Kind	Returns CRObjectKind object which specifies what "kind" of object (that is, box, cross-tab, field, etc.).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2

Property	*Description*	*Read/Write*
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

FieldValue Object

The FieldValue Object is only available through the *Window Object, Page 377*, event *DrillOnDetail, Page 380*, used in Visual Basic 5.0. When a user drills 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.

FieldValue Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Name	Returns the name of the field.	Read only
Parent	Reference to the Parent object (**Window Object** (page 377)).	Read only
Report	Reference to **Report Object** (page 347).	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 (that is, Bold, line style, etc.). A Font Object is obtained from the Font property of the *FieldObject Object, Page 314*. Options, such as font name, defined by this object are dependent upon the printer driver selected for the report.

> ***Note: You must always use a 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 281***.**

Font Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 by using RGB (that is, RGB (255, 0, 0 for red) or by specifying CRColor (see table below).	Read/Write
	Constant	***Value***
	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
Property	***Description***	***Read/Write***
Italic	Returns/Sets Boolean value indicating whether or not the font is italicized.	Read/Write
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 (**FieldObject Object** (page 314)).	Read only
Report	Reference to **Report Object** (page 347).	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
	Constant	***Value***
	crFWBold	700
	crFWDontCare	0
	crFWExtraBold	800
	crFWExtraLight	200
	crFWHeavy	900
	crFWLight	300
	crFWMedium	500
	crFWNormal	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.

FormulaFieldDefinition Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
FormulaFieldName	Returns the formula field name as it appears in the Formula Field list on the Formula Tab of the Insert Fields dialog box (for example, NewExampleFormula).	Read only
Kind	Returns CRFieldKind (see table below), which specifies what "kind" of field (that is, database, summary, formula, etc.).	Read only
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5

	crParameterField	6
	crSpecialVarField	4
	crSummaryField	3
Property	***Description***	***Read/Write***
Name	Returns the name of the formula field as it would be displayed (referenced) in the report (that is, {@NewExampleFormula}).	Read only
NumberOfBytes	Returns the number of bytes required to store the field data in memory.	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
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 **Check** (page 319), to check formulas. If you generate a report with an invalid formula, you may receive and exception error.	Read/Write
ValueType	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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
	crOleField	20
	crPersistentMemoField	14

crPictureField	19
crStringField	12
crTimeField	11
crTransientMemoField	13
crUnknownField	22

FormulaFieldDefinition Methods

Check, Page 319

Check

The Check method checks formula for errors (that is, syntax errors).

Syntax

```
object.Check
```

Returns

Returns a Boolean value,

- TRUE if formula is valid.
- FALSE if formula contains errors.

Remarks

You can use LastErrorCode and LastErrorString properties from the *Report Object, Page 347*, 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 317*, in the collection by using the Item property.

FormualFieldDefinitions Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of formula field definitions in the collection.	Read only

Property	*Description*	*Read/Write*
Item	Returns **FormulaFieldDefinition Object** (page 317). 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 that they were added to the report.	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	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 289*.

GlobalOptions Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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
Parent	Reference to the Parent object (**Application Object** (page 289)).	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 chart in your report (that is, graph data type -- group, detail or graph display type -- bar, pie, etc.).

GraphObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), this object is associated with.	Read only
ColumnGroupNumber	Returns/Sets the group number that the column of the graph gets its data from (that is, 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 **CrossTabObject Object** (page 298), 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
DataFields	Returns the FieldDefinitions object that represents the data fields of the graph (that is, 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
	Constant	***Value***
	crCrossTabGraph	2
	crDetailGraph	1
	crGroupGraph	0
Property	***Description***	***Read/Write***
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 (that is, for stacked bar graph). This property applies to only cross-tab graphs.	Read/Write
	Constant	***Value***
	crGDColumnsOnly	1
	crGDMixedColumnRow	3
	crGDMixedRowColumn	2
	crGDRowsOnly	0
	crGDUnknownDirection	20

Property	*Description*	*Read/Write*
DisplayType	Returns/Sets CRGraphDisplayType (see table below) indicating which graph representation type used for the graph (that is, SideBySideGraph, PieGraph, etc.).	Read/Write
	Constant	***Value***
	crAreaGraph	120
	crFaked3DPercentBarGraph	6
	crFaked3DSideBySideBarGraph	4
	crFaked3DStackedBarGraph	5
	crLineGraph	80
	crMultiPieGraph	42
	crPercentBarGraph	3
	crPieGraph	40
	crProportionalMultiPieGraph	43
	crSideBySideGraph	0
	crStackedBarGraph	2
	crThreedBarGraph	160
	crUnkownTypeGraph	1000
	crUserDefinedGraph	500
Property	***Description***	***Read/Write***
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 that are being graphed.	Read/Write
Kind	Returns CRObjectKind object(see table below), which specifies what "kind" of object (that is, box, cross-tab, field, etc.).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3

	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Property	***Description***	***Read/Write***
MaxValue	Returns/Sets the maximum value that will appear in the graph. Any graph values above this value are not graphed.	Read/Write
MinValue	Returns/Sets the minimum value that will appear in the graph. Any graph values below this value are not graphed.	Read/Write
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	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
SeriesTitle	Returns/Sets the title of the series which is being graphed.	Read/Write
ShowDataValue Enabled	Returns/Sets Boolean value indicating whether or not to display the numeric value associated with each riser on the graph. If set to TRUE (1), a value appears 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
ShowVerticalBar Enabled	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 **SummaryFieldDefinition Object** (page 367), 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 (that is, keep group together, sort direction, etc.). A GroupAreaOptions Object is obtained from the GroupOptions property of the *Area Object, Page 293*.

GraphAreaOptions Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Condition	Returns/Sets CRGroupCondition (see table below) indicating when grouping should occur for Boolean or Date groups (that is, for Boolean groups, group on change to TRUE; for date groups, group dates weekly, etc.).	Read/Write
	Constant	***Value***
	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
Property	***Description***	***Read/Write***
ConditionField	Returns/Sets **DatabaseFieldDefinition Object** (page 300), specifying the field that grouping occurs on.	Read/Write

DiscardOtherGroups	Returns/Sets Boolean value indicating if "other" groups in a group sort are discarded (that is, 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 (that is, across page breaks).	Read/Write
NumberOfTopOr BottomGroups	Returns/Sets the number of groups for a TopN group sort.	Read/Write
Parent	Reference to the Parent object (**Area Object** (page 293)).	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 **Report Object** (page 347).	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
	Constant	***Value***
	crAscendingOrder	1
	crDescendingOrder	0
	crOriginalOrder	2
	crSpecifiedOrder (Read only -- this value cannot be set.)	3
Property	***Description***	***Read/Write***
TopOrBottomNGroups	Returns/Sets the value of CRTopBottomNGroupSortOrder (see table below) indicating which section of groups will be shown (that is, TopN groups, BottomN groups, all groups, etc.).	Read/Write
	Constant	***Value***
	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 (that is, number of group, value type, etc.). A GroupNameFieldDefinition Object is obtained from the Field property of the *FieldObject Object, Page 314*, when the specified field is a group name field.

GroupNameFieldDefinition Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (that is, database, summary, formula, etc.).	Read only
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5
	crParameterField	6
	crSpecialVarField	4
	crSummaryField	3
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 (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
ValueType	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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

LineObject Object

The LineObject Object represents a line drawn on a report. This object provides properties for getting information for lines on a report.

LineObject Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that 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
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Property	***Description***	***Read/Write***
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	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.

OLEObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Property	***Description***	***Read/Write***
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Page Object

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. Unless you are experienced with the Crystal Report Engine Object Model, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code, in your Active Server Pages, to control 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.

Page Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (**PageGenerator Object** (page 333)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Page Object Methods

RenderEPF

The RenderEPF method returns a variant that contains EPF code for the report page.

Syntax

```
variantdata = object.RenderEPF (ResultType as CRRenderResultType)
```

Parameters

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

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
	crPlainPageStyle	0
	crToolbarPageStyle	1
ToolbarStyle	Specifies CRHTMLToolbarStyle (see table below) indicating the style of the toolbar to be used.	
	Constant	***Value***
	crToolbarRefreshButton	1
	crToolbarSearchBox	2
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.	
	Constant	***Value***
	crBSTRType	8
	crUISafeArrayType	8209

PageEngine Object

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. Unless you are experienced with the Crystal Report Engine Object Model, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code, in your Active Server Pages, to control 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.

PageEngine Object 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

PageEngine Object Methods

CreatePageGenerator, Page 331

RenderTotallerETF, Page 331

RenderTotallerHTML, Page 332

CreatePageGenerator

The CreatePageGenerator method returns a *PageGenerator Object, Page 333*, allowing you to get pages from the view of the report, specified by the GroupPath parameter.

Syntax

```
PgGenObject = object.CreatePageGenerator (GroupPath)
```

Parameters

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

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 by using strings or arrays.	
	Constant	***Value***
	crBSTRType	8
	crUISafeArrayType	8209
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 (that is, 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

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.	
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 by using strings or arrays.	
	Constant	***Value***
	crBSTRType	8
	crUISafeArrayType	8209

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 (that is, 0 = first group in Group #1 and 1 = second group in Group #2).
StartingChild Number	Long

PageGenerator Object

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. Unless you are experienced with the Crystal Report Engine Object Model, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code, in your Active Server Pages, to control 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, Page 328*, as they are requested and allows options for manipulating the report as a whole.

PageGenerator Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
GroupPath	Returns the GroupPath parameter set by **CreatePageGenerator** (page 331).	Read only
Pages	Returns **Pages Collection** (page 335), which is a collection of pages in the report.	Read only
Parent	Reference to the Parent object (**PageEngine Object** (page 330)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

PageGenerator Object Methods

DrillOnGraph, Page 334

GetPageNumberForGroup, Page 334

SearchForText, Page 334

DrillOnGraph

The DrillOnGraph method creates a new *PageGenerator Object, Page 333*, that results from drilling down on the specified point in a graph on the given page.

Syntax

```
pgGenObject = object.DrillOnGraph (PageNumber, XOffset, YOffset)
```

Parameters

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

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

Direction	Specifies CRSearchDirection (see table below) indicating the direction to sort in.	
	Constant	***Value***
	crAscendingOrder	1
	crDescendingOrder	0
	crOriginalOrder	2
PageNumber	Specifies the page to 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

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. Unless you are experienced with the Crystal Report Engine Object Model, you should allow the Crystal Design-Time ActiveX Control to generate VBScript code, in your Active Server Pages, to control 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*, *Page 328*. Access a specific Page Object in the collection using the Item property.

Pages Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of page objects in the collection.	Read only
Item	Returns **Page Object** (page 328). Item has an index parameter that is a numeric, 1-based index (that is, Item (1)).	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Remarks

Instead of using the Item property as shown, you can reference a page directly (Pages (1), for example).

PageSetup Object

The PageSetup Object provides properties for retrieving information and setting options for page setup in a report (that is, paper size, margins, etc.) for the current printer. A PageSetup Object is obtained from the PageSetup property of the *Report Object, Page 347*. If the current printer is changed with the ReportObject SelectPrinter method, then the values in PageSetup Object properties may change.

PageSetup Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
BottomMargin	Returns/Sets value specifying the bottom margin, in twips. See Remarks below.	Read/Write
LeftMargin	Returns/Sets value specifying the left margin, in twips. See Remarks below.	Read/Write
PaperOrientation	Returns/Sets CRPaperOrientation, using one of the following constants, indicating the paper orientation. See Remarks below.	Read/Write
	Constant	***Value***
	crDefaultPaperOrientation (Read Only)	0
	crLandscape	2
	crPortrait	1
PaperSize	Returns/Sets CRPaperSize, using one of the following constants, 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. See Remarks below.	Read/Write
	Constant	***Value***
	crDefaultPaperSize (Read Only)	0
	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
crPaperLegal	5
crPaperLetter	1
crPaperLetterSmall	2
crPaperNote	18
crPaperQuarto	15
crPaperStatement	6
crPaperTabloid	3

Property	*Description*	*Read/Write*
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
RightMargin	Returns/Sets value specifying the right margin, in twips. See Remarks below.	Read/Write
TopMargin	Returns/Sets value specifying the top margin, in twips. See Remarks below.	Read/Write

Remarks

- PageSetup properties are retrieved and set for the current printer. If the current printer is changed using ReportObject method SelectPrinter, then the properties of PageSetup Object, for example, PaperOrientation, can change.
- Properties BottomMargin, LeftMargin, RightMargin, and TopMargin values are 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.
- Properties BottomMargin, LeftMargin, RightMargin, and TopMargin are set to -1 by default. Unless otherwise specified, the default margins set in the report are used.
- Properties PaperOrientation and PaperSize default constants (crDefaultPaperOrientation and crDefaultPaperSize) are Read Only. The other enums listed below each property can be passed when setting a value.

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 (that is, current value, default value, etc.). A ParameterFieldDefinition Object is obtained from the Field property of the *Area Object, Page 293* when the specified field is a parameter field.

ParameterFieldDefinition Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
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
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5
	crParameterField	6
	crSpecialVarField	4
	crrSQLExpressionField	1
	crSummaryField	3
Name	Returns the name of the parameter field as it appears in the Parameter Field list on the Parameter Tab of the Insert Fields dialog box (for example, NewExampleParameter).	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
ParameterField Name	Returns the name of the parameter field as it is displayed (referenced) in the report (that is, {?ExampleParameter}).	Read only
Parent	Reference to the Parent object (**Report Object** (page 347)).	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 **Report Object** (page 347).	Read only
ReportName	Returns the subreport name if the parameter is part of a subreport; otherwise, a null is returned.	Read only

ValueType	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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
	crOleField	20
	crPersistentMemoField	14
	crPictureField	19
	crStringField	12
	crTimeField	11
	crTransientMemoField	13
	crUnknownField	22

ParameterFieldDefinition Object Methods

SetCurrentValue, Page 340

SetDefaultValue, Page 342

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

CurrentValue	Specifies the value that 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 (that is, 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
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

DefaultValue	Specifies the value that 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.

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
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 338*, in the collection by using the Item property.

ParameterFieldDefinitions Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of parameter fields in the collection.	Read only
Item	Returns **ParameterFieldDefinition Object** (page 338), 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 that 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 (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Remarks

Instead of using the Item property as shown, you can reference a parameter field directly (for example, ParameterFieldDefinitions(1)).

PrinterInfo Object

The PrinterInfo Object provides properties for retrieving information and setting options for the specified printer used to print a report (that is, driver name, port name, etc.). A PrinterInfo Object is obtained from the PrinterInfo property of the *Report Object, Page 347*.

PrinterInfo Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (**Report Object** (page 347)).	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 **Report Object** (page 347).	Read only

PrintingStatus Object

The PrintingStatus Object provides properties for retrieving information and setting options for the printing status of a report (that is, number of pages, latest page to be printed, etc.). A PrintingStatus Object is obtained from the PrintingStatus property of the *Report Object, Page 347*.

PrintingStatus Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
LastestPageNumber	Returns the number of the last page that is 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 ShowNextPage, Page 375, 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 (**Report Object** (page 347)).	Read only

Property	*Description*	*Read/Write*
Progress	Returns CRPrintingProgress (see table below) indicating the status of printing report (that is, cancelled, in progress, not started, etc.). Corresponds to PEJobInfo and PEGetJobStatus *(Seagate Crystal Reports Technical Reference Guide)*.	Read only
	Constant	***Value***
	crPrintingCancelled	5
	crPrintingCompleted	3
	crPrintingFailed	4
	crPrintingHalted	6
	crPrintingInProgress	2
	crPrintingNotStarted	1
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 ShowNextPage, Page 375, 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 **Report Object** (page 347).	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 (that is, 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 378*, 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 347*.

PrintWindowOptions Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 whether the preview window includes the Group Tree. You **must** also set the HasGroupTree property to TRUE in the **ReportOptions Object** (page 356), to have the Group Tree show in the preview window.	Read/Write
HasNavigation Controls	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
HasPrintSetup Button	Returns/Sets Boolean value indicating whether the preview window has a Print Setup button.	Read/Write
HasProgress Controls	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 (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
TrackCursorInfo	Returns **TrackCursorInfo Object** (page 371).	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 289*. When a Report Object gets 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.

Report Object Properties

Properties	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Areas	Returns **Areas Collection** (page 295), 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 **Database Object** (page 299), which represents the database used in the report.	Read only
DialogParent Window	Sets the parent window of report dialog boxes. If not set, the parent window is the preview window.	Write only
EventInfo	Returns **EventInfo Object** (page 309).	Read only
ExportOptions	Returns **ExportOptions Object** (page 309).	Read only
FormulaFields	Returns FormulaFieldDefinitions Collection, Page 319, a collection of all the named FormulaFieldDefinitions defined in the Report.	Read only
GroupSelection Formula	Returns/Sets the group selection formula.	Read/Write
GroupSortFields	Returns SortFields Collection, Page 363, a collection of group sort fields.	Read only
HasSavedData	Returns Boolean value indicating whether report includes saved data.	Read only
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

Properties	*Description*	*Read/Write*
Options	Returns **ReportOptions Object** (page 356).	Read only
PageEngine	Returns **PageEngine Object** (page 330).	Read only
PageSetup	Returns **PageSetup Object** (page 336).	Read only
ParameterFields	Returns **ParameterFieldDefinitions Collection** (page 343), 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 (that is, if the main report has 3 parameters and if 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
Parameter PromptingEnabled	Returns/Sets Boolean value indicating whether prompting dialog box for parameter field information is used. • 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 (**Application Object** (page 289)).	Read only
PrintDate	Returns/Sets the print date for the report. By default, the current date will be used.	Read/Write
PrinterInfo	Returns **PrinterInfo Object** (page 343).	Read only
PrintingStatus	Returns **PrintingStatus Object** (page 344).	Read only
PrintWindow Options	Returns **PrintWindowOptions Object** (page 345).	Read only
ProgressDialog Enabled	Sets Boolean value indicating whether or not to enable the progress dialog box when printing or exporting.	Write only
RecordSelection Formula	Returns/Sets record selection formula.	Read/Write
RecordSortFields	Returns **SortFields Collection** (page 363), a collection of record sort fields.	Read only
ReportSummary Info	Returns **ReportSummaryInfo Object** (page 358).	Read only
Sections	Returns Sections Collection, Page 361, a collection of all the sections in the report.	Read only
SQLQueryString	Returns/Sets the SQL query used in the report.	Read/Write
SummaryFields	Returns SummaryFieldDefinitions Collection, Page 369, for group and report summaries (cross-tab summaries not available when using this property).	Read only

Report Object 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

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.

Constant	***Value***
crGCAnually	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
SortDirection	Specifies CRSortDirection (see table below) indicating the sort direction for the group (that is, ascending, descending).	
	Constant	***Value***
	crAscendingOrder	1
	crDescendingOrder	0
	crOriginalOrder	2

CancelPrinting

The CancelPrinting method cancels the printing of a report. It corresponds to PECancelPrintJob *(Seagate Crystal Reports Technical Reference Guide)* 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 *(Seagate Crystal Reports Technical Reference Guide)* of the Crystal Report Engine.

Syntax

```
object.DiscardSavedData
```

Export

The Export method exports reports to a format and destination specified with *ExportOptions Object, Page 309*.

Syntax

```
object.Export PromptUser
```

Parameters

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 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 347*, corresponding to the named subreport. It corresponds to PEOpenSubreport *(Seagate Crystal Reports Technical Reference Guide)* of the Crystal Report Engine.

Syntax

```
set reportvar = object.OpenSubreport (SubreportName)
```

Parameter

SubreportName	Specifies the file name of the subreport to be opened.

Returns

Report

Preview

The Preview method reviews the report and returns the *View Object, Page 373*.

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

Title (Optional)	Specifies the string that contains the title that 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 PEOutputToWindow *(Seagate Crystal Reports Technical Reference Guide)*.
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 373*, which represents the view created by previewing.

PrintOut

The PrintOut method prints out the specified pages of the report to the printer selected by using the *SelectPrinter, Page 353*, 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

PromptUser (Optional)	Specifies Boolean value indicating if the user should be prompted for printer options.
NumberOfCopies (Optional)	Specifies the number of report copies that you want printed.
Collated (Optional)	Specifies Boolean value specifying whether or not you want the report copies collated.
StartPageNumber (Optional)	Specifies the first page that you want printed.
StopPageNumber (Optional)	Specifies the last page that you want printed.

ReadRecords

The ReadRecords method reads records in the report.

Syntax

```
object.ReadRecords
```

SelectPrinter

The SelectPrinter method selects a different printer for the report. The values associated with the properties of PageSetup Object may change if a call to SelectPrinter method is made.

Syntax

```
object.SelectPrinter DriverName, PrinterName, PortName
```

Parameters

DriverName	String that contains the name of the printer driver for the selected printer. See DriverName property of **PrinterInfo Object** (page 343) for more information.
PrinterName	String that contains the printer name for the selected printer. See PrinterName property of **PrinterInfo Object** (page 343) for more information.
PortName	String that contains the port name for the port to which the selected printer is attached. See PortName property of **PrinterInfo Object** (page 343) for more information.

Report Object Events

ReadingRecords

The ReadingRecords event occurs when records are read (that is, 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

Cancelled	Specifies Boolean value indicating that record reading was cancelled.
Done	Specifies Boolean value indicating 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 by using the ReadingRecordsEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information on events, see *Object Model Events, Page 282*.

Start

The Start event occurs when the report starts printing or previewing.

Syntax

```
Event Start (Destination As CRPrintingDestination,
useDefault As Boolean)
```

Parameters

Destination	Specifies CRPrintingDestination (see table below) indicating what "destination" the report will be printed to (that is, printer, window, exported, etc.).	
	Constant	***Value***
	crFromQuery	4
	crToExport	3
	crToNoWhere	0
	crToPrinter	2
	crToWindow	1
useDefault	Specifies Boolean value indicating if default behavior should occur after user's event code has run (that is, should the report start to print or preview?).	

Remarks

This event is enabled by using the StartStopEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

Stop

The Stop event that occurs when report stops printing/previewing.

Syntax

```
Event Stop (Destination As CRPrintingDestination,
Status As CRPrintingProgress)
```

Parameters

Destination	Specifies CRPrintingDestination (see table below) indicating what "destination" the report will be printed to (that is, printer, window, exported, etc.).	
	Constant	***Value***
	crFromQuery	4
	crToExport	3
	crToNoWhere	0
	crToPrinter	2
	crToWindow	1
Status	Specifies CRPrintingProgress (see table below) indicating the progress of the report print job (that is, in progress, halted, cancelled, etc.).	
	Constant	***Value***
	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, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

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.

ReportObjects Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
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 **BlobFieldObject Object** (page 296), **FieldObject Object** (page 314),**TextObject Object** (page 370), **OLEObject Object** (page 327), **CrossTabObject Object** (page 298), **SubreportObject Object** (page 366), **BoxObject Object** (page 297), **GraphObject Object** (page 320), or a **LineObject Object** (page 327). 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 (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	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.

ReportOptions Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), this object is associated with.	Read only
CaseInsensitiveSQL Data	Returns/Sets Boolean value indicating if SQL data is case-sensitive.	Read/Write

ConvertDateTimeTo Date	Returns/Sets CRConvertDateTimeType (see table below) indicating what datetime fields are to be converted to (that is, string, time, no conversion, etc.).	Read/Write
	Constant	***Value***
	crConvertDateTimeToDate	1
	crConvertDateTimeToString	0
	crKeepDateTimeType	2
Property	***Description***	***Read/Write***
ConvertNullFieldTo Default	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 **PrintWindowOptions Object** (page 345), to have the Group Tree show in the preview window.	Read/Write
MorePrintEngine ErrorMessages	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 by using this property.	Read only
Report	Read only	Read only
SaveDataWithReport	Returns/Sets Boolean value indicating if data is saved with the report.	Read/Write
SaveSummariesWith Report	Returns/Sets Boolean value indicating if summary values (that is, 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
	Constant	***Value***
	crPageWidth	1
	crWholePage	2

ReportSummaryInfo Object

The ReportSummaryInfo Object provides properties for retrieving information and setting options for summary information included in a report (that is, author, title, etc.). A ReportSummaryInfo Object is obtained from the ReportSummaryInfo property of the *Report Object, Page 347*.

ReportSummaryInfo Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	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.

Section Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 (that is, if the first section in an area, the number returned is 1).	Read only

Property	*Description*	*Read/Write*
Options	Returns **SectionOptions Object** (page 359).	Read only
Parent	Reference to the Parent object (**Area Object** (page 293)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
ReportObjects	Returns **ReportObjects Collection** (page 355), 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 that 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 (that is, background color, new page after, etc.). A SectionOptions Object is obtained from the Options property of the *Section Object, Page 358*.

Each report area is comprised of sections that can be formatted independently of each other using the SectionOptions object. These section options are combined with the options set by using the *AreaOptions Object, Page 294*, to make up the final appearance of a section and an 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.

SectionOptions Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 by using OLE colors (that is, by using RGB function).	Read/Write
	Constant	***Value***
	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
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
KeepTogether Formula	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
NewPageAfter Formula	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

NewPageBefore Formula	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 (**Section Object** (page 358)).	Read only
PrintAtBottomOf Page	Returns/Sets Boolean indicating if the section is printed at the bottom of the page.	Read/Write
PrintAtBottomOf PageFormula	Returns/Sets string specifying formula for conditionally printing section at the bottom of the page.	Read/Write
Report	Reference to **Report Object** (page 347).	Read only
ResetPageNumber After	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
SuppressIfBlank Formula	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
UnderlaySection Formula	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 347*, or *Area Object, Page 293*. The Sections Collection is a collection of section objects. Access a specific Section Object in the collection by 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.

Sections Collection Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only

Property	*Description*	*Read/Write*
Count	Returns the number of section in the collection.	Read only
Item	Returns **Section Object** (page 358). Item has an index parameter that can be either a string reference to the area section (that is, for areas with one section: "RH", "PH", "GHn", "D", "GFn", "PF", or "RF") or a numeric, 1-based index (that is, 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 (that is, "Da", "Db", etc.).	Read only
Parent	Reference to the Parent object (**Report Object** (page 347), or **Area Object** (page 293)).	Read only
Report	Reference to **Report Object** (page 347).	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.

SortField Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), this object is associated with.	Read only
Field	Returns/Sets **DatabaseFieldDefinition Object** (page 300), which specifies the sort field used in the report.	Read/Write
Parent	Reference to the Parent object (**Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read/Write
SortDirection	Returns/Sets CRSortDirection (see table below) indicating the direction the group/records are sorted in.	Read/Write
	Constant	***Value***
	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 that can be either a record sort field, or a group sort field. Access a specific SortField Object in the collection by using the Item property.

SortFields Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of sort fields in the collection.	Read only
Item	Returns **SortField Object** (page 362). Item has an index parameter that is a numeric, 1-based index (that is, Item (1)). The sort fields in the collection are indexed in the order that they were added as sort fields in the report.	Read only
Parent	Reference to the Parent object **Report Object** (page 347).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Remarks

Instead of using the Item property as shown, you can reference a sort field directly (for example, SortFields(1)).

SortFields Collection Methods

Add

The Add method adds a record or group sort field.

Syntax

```
object.Add SortDirection, Field
```

ParametersL

SortDirection	Specifies CRSortDirection (see table below) indicating the direction in which the field data should be sorted (that is, ascending, descending, etc.).	
	Constant	***Value***
	crAscendingOrder	1
	crDescendingOrder	0
	crOriginalOrder	2
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
```

Parameter

Index	Specifies the 1-based number indicating which sort field in the collection should be deleted. The first sort field is sort field 1.

SpecialVarFieldDefinition Object

The SpecialVarFieldDefinition Object provides properties for retrieving information and setting options for a special field found in your report (that is, last modification date, print date, etc.). A SpecialVarFieldDefinition Object is obtained from the Field property of the *FieldObject Object, Page 314*.

SpecialVarFieldDefinition Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that 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
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5
	crParameterField	6
	crSpecialVarField	4
	crSummaryField	3
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 **Report Object** (page 347).	Read only
Report	Reference to **Report Object** (page 347).	Read only
SpecialVarType	Returns CRSpecialVarType (see table below) value indicating the type of special field (that is, ReportTitle, PageNumber, etc.).	Read only
	Constant	***Value***
	crSVTDataDate	7
	crSVTDataTime	8
	crSVTGroupNumber	6
	crSVTModificationDate	2
	crSVTModificationTime	3
	crSVTPageNumber	5
	crSVTPrintDate	0
	crSVTPrintTime	1
	crSVTRecordNumber	4
	crSVTReportComments	11
	crSVTReportTitle	10
	crSVTTotalPageCount	9
Property	***Description***	***Read/Write***
ValueType	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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
crOleField	20
crPersistentMemoField	14
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 (that is, name, etc.).

Subreport Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only

Kind	Returns CRObjectKind (see table below) that specifies what "kind" of object (that is, box, cross-tab, field).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7
	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Name	Returns the name of the subreport.	Read only
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	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.

SummaryFieldDefinition Object Properties

Property	***Description***	***Read/Write***
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
CrossTabObject	Returns **CrossTabObject Object** (page 298), 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 **Area Object** (page 293), 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 when using this property.	Read only
ForCrossTab	Returns Boolean value indicating if summary is used in a cross-tab.	Read only

HeaderArea	Returns **Area Object** (page 293), indicating the Group/Report Header area of the summary. If the summary is from a cross-tab you will get an exception when using this property.	Read only
Kind	Returns CRFieldKind, which specifies what "kind" of field (database, summary, formula, etc.).	Read only
	Constant	***Value***
	crDatabaseField	1
	crFormulaField	2
	crGroupNameField	5
	crParameterField	6
	crSpecialVarField	4
	crSummaryField	3
Property	***Description***	***Read/Write***
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 **Report Object** (page 347).	Read only
Report	Reference to **Report Object** (page 347).	Read only
SummaryType	Returns value of type CRSummaryType (see table below) indicating the type of summary (that is, sum, average, etc.).	Read only
	Constant	***Value***
	crSTAverage	1
	crSTCount	6
	crSTDisctintCount	9
	crSTMaximum	4
	crSTMinimum	5
	crSTPopStandardDeviation	8
	crSTPopVariance	7
	crSTSampleStandardDeviation	3
	crSTSampleVariance	2
	crSTSum	0

ValueType	Returns CRFieldValueType (see table below), which specifies the "type" of value found in the field.	Read only
	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
	crOleField	20
	crPersistentMemoField	14
	crPictureField	19
	crStringField	12
	crTimeField	11
	crTransientMemoField	13
	crUnknownField	22

SummaryFieldDefinitions Collection

The SummaryFieldDefinitions Collection is a collection of summary field definitions. They can be from either *CrossTabObject Object, Page 298*, or *Report Object, Page 347*. Access a specific SummaryFieldDefinition Object in the collection by using the Item property.

SummaryFieldDefinitions Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Count	Returns the number of summary fields in the collection.	Read only
Item	Returns **SummaryFieldDefinition Object** (page 367). Item has an index parameter that is a numeric, 1-based index (that is, Item (1)). The items in the collection are indexed in the order that they were added to the report.	Read only
Parent	Reference to the Parent object **CrossTabObject Object** (page 298),or **Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

Remarks

Instead of using the Item property as shown, you can reference a summary field directly, for example, SummaryFieldDefinitions (1).

TextObject 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.

TextObject Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
Kind	Returns CRObjectKind (see table below) that specifies what "kind" of object (that is, box, cross-tab, field).	Read only
	Constant	***Value***
	crBlobFieldObject	9
	crBoxObject	4
	crCrossTabObject	8
	crFieldObject	1
	crGraphObject	7

	crLineObject	3
	crOleObject	6
	crSubreportObject	5
	crTextObject	2
Parent	Reference to the Parent object (**Section Object** (page 358)).	Read only
Report	Reference to **Report Object** (page 347).	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.

TrackCursorInfo Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that this object is associated with.	Read only
DetailAreaCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor is 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
DetailAreaField Cursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor is 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
GraphCursor	Returns/Sets CRTrackCursor (see table below GroupAreaFieldCursor property) indicating what type of cursor is 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 is 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
GroupAreaField Cursor	Returns/Sets CRTrackCursor (see table below) indicating what type of cursor is 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
	Constant	***Value***
	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
Property	***Description***	***Read/Write***
Parent	Reference to the Parent object **PrintWindowOptions Object** (page 345)).	Read only
Report	Reference to **Report Object** (page 347).	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 347*, 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.

View Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 ShowNextPage, Page 375, 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 **Window Object** (page 377)).	Read only
Report	Reference to **Report Object** (page 347).	Read only

View Object Methods

Close, Page 374

Export, Page 374

NextMagnification, Page 374

PrintOut, Page 374

ShowFirstPage, Page 374

ShowLastPage, Page 375

ShowNextPage, Page 375

ShowNthPage, Page 375

ShowPreviousPage, Page 375

ZoomPreviewWindow, Page 375

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 (that is, 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)
```

Parameter

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)
```

Parameter

ZoomLevel	Specifies the percentage of magnification for the view (for example, 50%, 75%, etc.).

Views Collection

The Views Collection is a collection of views for a report. A *View Object, Page 373*, 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)
```

Views Collection Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 **View Object** (page 373). Item has an index parameter that is a numeric, 1-based index (that is, 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 **Window Object** (page 377)).	Read only
Report	Reference to **Report Object** (page 347).	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.

Window Object Properties

Property	*Description*	*Read/Write*
Application	Returns a reference to the **Application Object** (page 289), that 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 **Report Object** (page 347)).	Read only
Report	Reference to **Report Object** (page 347).	Read only
Views	Returns **Views Collection** (page 376), a collection of views in the window.	Read only
WindowHandle	Returns the handle of the window.	Read only

Window Object Methods

Close Method

The Close method closes the window, and consequently, all views in the window.

Syntax

```
object.Close
```

Window Object Events

ActivatePrintWindow

The ActivatePrintWindow event occurs when the print window is activated (that is, received focus from another window).

Syntax

```
Event ActivatePrintWindow ()
```

Remarks

This event is enabled by using the ActivatePrintWindowEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

CancelButtonClicked

The CancelButtonClicked event occurs when the Cancel button is clicked.

Syntax

```
Event CancelButtonClicked (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, 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 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

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

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 (that is, if you do not want the view/preview window to close).

Remarks

Before enabling this event, the HasCloseButton property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

ClosePrintWindow

The ClosePrintWindow event occurs when the print window is closed.

Syntax

```
Event ClosePrintWindow (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want the print window to close).

Remarks

This event is enabled by using the ClosePrintWindowEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

DeactivatePrintWindow

The DeactivatePrintWindow event occurs when the print window is deactivated (that is, when any other window takes the focus from the print window).

Syntax

```
Event DeactivatePrintWindow ()
```

Remarks

This event is enabled by using the ActivatePrintWindowEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

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

FieldValues	Specifies an array of FieldValue objects (see *FieldValue Object, Page 315*). 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 that 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.

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, 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 345*, must be set to TRUE. For more information, see *Object Model Events, Page 282*.

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

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.	
	Constant	***Value***
	crDrillOnGraph	2
	crDrillOnGroup	0
	crDrillOnGroupTree	1
useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, 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 345*, must be set to TRUE. This event is enabled by using the GroupEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

ExportButtonClicked

The ExportButtonClicked event occurs when the Export button is clicked.

Syntax

```
Event ExportButtonClicked (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want the Export dialog box to appear).

Remarks

Before enabling this event, the HasExportButton property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

FirstPageButtonClicked

The FirstPageButtonClicked event occurs when the button, that navigates through the report to the first page, is clicked.

Syntax

```
Event FirstPageButtonClicked (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want to go to the first page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

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

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 345*, and the HasGroupTree property of the *ReportOptions Object, Page 356* must be set to TRUE. This event is enabled by using the GroupEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

LastPageButtonClicked

The LastPageButtonClicked event occurs when the button, that navigates through the report to the last page, is clicked.

Syntax

```
Event LastPageButtonClicked (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want to go to the last page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

NextPageButtonClicked

The NextPageButtonClicked event occurs when the button, that navigates through the report to the next page, is clicked.

Syntax

```
Event NextPageButtonClicked (useDefault As Boolean)
```

Parameter

useDefault	Specifies Boolean value indicating whether the default values should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want to go to the next page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

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

useDefault	Specifies Boolean value indicating whether the default values should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want to go to the previous page).

Remarks

Before enabling this event, the HasNavigationControls property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

PrintButtonClicked

The PrintButtonClicked event occurs when the Print button is clicked.

Syntax

```
Event PrintButtonClicked (useDefault As Boolean)
```

Parameters

useDefault	Specifies Boolean value indicating whether the default settings should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want the Print dialog box to appear).

Remarks

Before enabling this event, the HasPrintButton property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

PrintSetupButtonClicked

The PrintSetupButtonClicked event occurs when the Print Setup button is clicked.

Syntax

```
Event PrintSetupButtonClicked (useDefault As Boolean)
```

Parameters

useDefault	Specifies Boolean value indicating whether the default setting should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, 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 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

RefreshButtonClicked

The RefreshButtonClicked event occurs when the Refresh button is clicked.

Syntax

```
Event RefreshButtonClicked (useDefault As Boolean)
```

Parameters

useDefault	Specifies Boolean value indicating whether the default setting should be used. Assign useDefault = FALSE if not using default behavior defined in CPEAUT (that is, if you do not want the report to refresh).

Remarks

Before enabling this event, the HasRefreshButton property of the *PrintWindowOptions Object, Page 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

SearchButtonClicked

The SearchButtonClicked event occurs when the Search button is clicked.

Syntax

```
Event SearchButtonClicked (SearchString As String,
useDefault As Boolean)
```

Parameters

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 (that is, 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 345*, must be set to TRUE. This event is enabled by using the PrintWindowButtonEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

ShowGroup

The ShowGroup event occurs when you click a group in the Group Tree.

Syntax

```
Event ShowGroup (GroupNameList, useDefault As Boolean)
```

Parameters

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 (that is, if you do not want the group that 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 345*, and the HasGroupTree property of the *ReportOptions Object, Page 356* must be set to TRUE. This event is enabled by using the GroupEventEnabled property of the *EventInfo Object, Page 309*, obtained through the EventInfo property of the *Report Object, Page 347*. For more information, see *Object Model Events, Page 282*.

ZoomLevelChanging

The ZoomLevelChanging event occurs when the zoom level of the report preview is changed.

Syntax

```
Event FirstPageButtonClicked (ZoomLevel As Integer)
```

Parameters

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 345*, must be set to TRUE. For more information, see *Object Model Events, Page 282*.

ERROR CODES

The following topics are discussed in this section.

Automation Server Error Codes

The Crystal Report Engine Automation Server generates the following error codes, as needed:

Value	*Error Code*	*Description*
30001	CPEAUT_STANDARD_OUTOFMEMORY	There is not enough memory to complete the action.
30002	CPEAUT_STANDARD_INVALDHANDLE	A handle that is invalid has been specified or does not exist.
30003	CPEAUT_STANDARD_INVALIDPOINTER	A pointer that is invalid has been specified 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 cannot 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 that is invalid or does not exist has been specified.
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 *RenderHTML, Page 329*, 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.

4 Crystal Class Library for NewEra Reference

What you will find in this chapter...

THE CRYSTAL NEWERA CLASS LIBRARY

Introduction

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 *class CRPEngine, Page 391*, 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 (for example, SQL connections, version information, etc.). More importantly, it is responsible for creating and managing all *class CRPEJob, Page 399*, objects. It is the CRPEJob object that allows you access to the attributes of a print job.

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 (that is, 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 398*, specifying the report file name to open. If successful, you will be returned a pointer to a CRPEJob object.

constructor CRPEngine::CRPEngine

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 397*.

Constructor Syntax

```
CRPEngine (v_open BOOLEAN : FALSE);
```

Parameter

v_open	Indicates whether or not the Crystal Report Engine should be opened when the CRPEngine object is created.

class CRPEngine Methods

The following methods are discussed in this section.

CRPEngine::CanClose

Use CRPEngine::CanClose to Check 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.

Syntax

```
CanClose () RETURNING BOOLEAN
```

Returns

- TRUE if the Crystal Report Engine can be closed safely.
- FALSE if the Crystal Report Engine is still busy.

Related Topics

CRPEngine::CanClose, Page 4

PECanCloseEngine in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::Close

This method closes the Crystal Report Engine.

Syntax

```
Close() RETURNING VOID
```

Related Topics

PECloseEngine in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::GetEngineStatus

This method may be used to get the current engine status.

Syntax

```
GetEngineStatus() RETURNING SMALLINT
```

Returns

One of the following values of the Status enumerated type indicating the current status of the Crystal Report Engine.

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.

CRPEngine::GetErrorCode

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.

Syntax

```
GetErrorCode () RETURNING SMALLINT
```

Returns

- Returns the current Crystal Report Engine **Error Codes** (page 468).
- Returns 0 if no error has occurred.

Related Topics

PEGetErrorCode in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::GetErrorText

This method returns a descriptive Crystal Report Engine error message.

Syntax

```
GetErrorText () RETURNING ixString
```

Returns

A ixString object containing the text for the current error code.

Related Topics

PEGetErrorText in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::GetNPrintJobs

This method returns the number of print job (CRPEJob) objects that currently exist for the current CRPEngine object.

Syntax

```
GetNPrintJobs () RETURNING SMALLINT;
```

Returns

- Returns the number of CRPEJob objects currently open.
- Returns -1 if an error occurs.

CRPEngine::GetVersion

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.

Syntax

```
GetVersion (versionRequested SMALLINT) RETURNING SMALLINT
```

Parameter

versionRequested	Specifies whether the version number of the Crystal Report Engine or the Report Engine DLL is being requested. Possible values are:	
	Value	***Meaning***
	PEP_GV_DLL	Returns the version of the DLL.
	PEP_GV_ENGINE	Returns the version of the Crystal Report Engine.

Returns

The version number of the currently used Crystal Report Engine or Report Engine DLL.

Related Topics

PEGetVersion in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::LogOffServer

This method closes an existing connection to a SQL server. Connection information is provided through the *class CRPELogOnInfo, Page 461*.

Syntax

```
LogOffServer(dllName CHAR (*), const CRPELogOnInfo logOnInfo) RETURNING
BOOLEAN
```

Parameters

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PELogOffServer in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::LogOnServer

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 **class CRPELogOnInfo** (page 461).

Syntax

```
LogOnServer (dllName CHAR(*), logOnInfo CRPELogOnInfo) RETURNING BOOLEAN;
```

Parameters

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 **class CRPELogOnInfo** (page 461).

Return Value

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PELogOnServer in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::LogOnSQLServerWithPrivateInfo

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. See Remarks below.

Syntax

```
LogOnSQLServerWithPrivateInfo (dllName CHAR (*), FOREIGN privateInfo)
RETURNING BOOLEAN
```

Parameters

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

The CRPEngine::LogOnSQLServerWithPrivateInfo method can only be used with database connections established by ODBC or Q+E Library 2.0. Any other database connections cannot 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEngine::Open

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.

Syntax

```
Open() RETURNING BOOLEAN
```

Returns

- TRUE if the Crystal Report Engine was opened successfully.
- FALSE if something went wrong.

Related Topics

constructor CRPEngine::CRPEngine, Page 391

PEOpenEngine in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::OpenJob

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.

Syntax

```
OpenJob (reportFileName CHAR(*)) RETURNING CRPEJob
```

Parameter

reportFileName	The name and path (if necessary) of the report file being opened for the specified print job.

Returns

CRPEJob object for the opened print job.

Related Topics

PEOpenPrintJob in the *Seagate Crystal Reports Technical Reference Guide*

CRPEngine::PrintReport

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.

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

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.

Returns

- Returns 0 if the call is successful.
- Returns an error code if the call fails.

Related Topics

PEPrintReport in the *Seagate Crystal Reports Technical Reference Guide*

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 398*, 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.

The class constructor is called automatically by CRPEngine::OpenJob ().

constructor CRPEJob::CRPEJob

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.

Constructor Syntax

```
CRPEJob (jobHandle SMALLINT);
```

Parameter

jobHandle	Handle to the job created by CRPEngine::OpenJob().

class CRPEJob Methods

The following methods, listed alphabetically, are discussed in this section.

class CRPEJob Methods A - F

class CRPEJob Methods G (Get...)

class CRPEJob Methods H - Z

CRPEJob::Cancel

Use Cancel to cancel processing of a print job. It can be tied to a control that allows the user to cancel a print job in progress.

Syntax

```
Cancel () RETURNING VOID
```

Related Topics

PECancelPrintJob in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::CheckFormula

Use CheckFormula 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.

Syntax

```
CheckFormula (*formulaName CHAR (*)) RETURNING BOOLEAN
```

Parameter

formulaName	The name of the formula that you need to check for errors.

Returns

- TRUE if the formula text is correct.
- FALSE if there is an error in the formula or if the call fails.

Related Topics

PECheckFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::CheckGroupSelectionFormula

Use CheckGroupSelectionFormula 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.

Syntax

```
CheckGroupSelectionFormula () RETURNING BOOLEAN
```

Returns

- TRUE if the formula text is correct.
- FALSE if there is an error in the formula or if the call fails.

Related Topics

PECheckGroupSelectionFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::CheckSelectionFormula

Use CheckSelectionFormula 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.

Syntax

```
CheckSelectionFormula () RETURNING BOOLEAN
```

Returns

- TRUE if the formula text is correct.
- FALSE if there is an error in the formula or if the call fails.

Related Topics

PECheckSelectionFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::Close

Use Close to close 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.

Syntax

```
Close () RETURNING VOID
```

Related Topics

PEClosePrintJob in the *Seagate Crystal Reports Technical Reference Guide2*

CRPEJob::CloseWindow

Use CloseWindow 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.

Syntax

```
CloseWindow () RETURNING VOID
```

Related Topics

PECloseWindow in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::DeleteNthGroupSortField

Use DeleteNthGroupSortField to delete 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.

Syntax

```
DeleteNthGroupSortField (sortFieldN SMALLINT) RETURNING BOOLEAN
```

Parameter

sortFieldN	Specifies the 0-based 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEDeleteNthGroupSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::DeleteNthSortField

Use DeleteNthSortField to delete 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.

Syntax

```
DeleteNthSortField (sortFieldN SMALLINT) RETURNING BOOLEAN
```

Parameter

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEDeleteNthSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::ExportPrintWindow

Use ExportPrintWindow to export 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.).

Syntax

```
ExportPrintWindow (toMail BOOLEAN) RETURNING BOOLEAN
```

Parameter

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

CRPEJob::ExportTo

Use ExportTo to set 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::StartJob, Page 450*.

Syntax

```
ExportTo (options CRPEExportOptions) RETURNING BOOLEAN
```

Parameter

options	Specifies a pointer to **class CRPEExportOptions** (page 452).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEExportTo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetErrorCode

Use GetErrorCode to retrieve 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.

Syntax

```
GetErrorCode () RETURNING SMALLINT
```

Returns

- Returns a Class Library **Error Codes** (page 468).
- Returns PEP_NOERROR if no error has occurred.

Related Topics

PEGetErrorCode in the *Seagate Crystal Reports Technical Reference Guide2*

CRPEJob::GetErrorText

Use GetErrorText to retrieve a descriptive error message for the print job.

Syntax

```
GetErrorText () RETURNING ixString
```

Returns

A ixString object containing the text for the current error.

Related Topics

PEGetErrorText, in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetExportOptions

Use GetExportOptions to prompt 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 **class CRPEExportOptions** (page 452). The *CRPEJob::ExportTo, Page 406*, function can then be used to set the print job destination using the information in this class.

Syntax

```
GetExportOptions () RETURNING BOOLEAN, CRPEExportOptions
```

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEExportOptions, Page 452

Related Topics

PEGetExportOptions in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetFormula

Use GetFormula to retrieve the formula text for the specified formula. CRPEJob::GetFormula is often used with *CRPEJob::SetFormula, Page 434,* to identify and then change an existing formula in response to a user selection at print time.

Syntax

```
GetFormula (formulaName CHAR(*)) RETURNING BOOLEAN, ixString
```

Parameter

formulaName	The name of the formula for which you want to retrieve the formula string.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

A ixString object containing the formula text.

Related Topics

PEGetFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetGraphData

Use GetGraphData to determine what data in your report is being used by a specified graph when the chart is created. This information includes which groups are used to create the rows and columns of the chart and which summary field in the report is used to set the values of the risers in the chart.

Syntax

```
GetGraphData (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,
CRPEGraphDataInfo
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart from which you want to retrieve data.
graphN	Specifies from which chart within the section you want to retrieve chart data information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.

Return Values

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEGraphDataInfo, Page 456

Related Topics

PEGetGraphTextInfo, in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetGraphOptions

Use GetGraphOptions to retrieve information about any of several chart options. These options include the minimum and maximum values that can appear on the chart, whether grid lines appear, whether risers are labeled, whether bar charts have horizontal or vertical bars, and whether a legend appears on the chart.

Syntax

```
GetGraphOptions (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,
CRPEGraphOptions
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart from which you want to retrieve options.
graphN	Specifies from which chart within the section you want to retrieve chart options information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEGraphOptions, Page 457

Related Topics

PEGetGraphOptionInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetGraphText

Use GetGraphText to determine what text will appear with a chart. A chart 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 charts).

Syntax

```
GetGraphText (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,
CRPEGraphTextInfo
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart from which you want to retrieve text.
graphN	Specifies from which chart within the section you want to retrieve chart text information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEGraphTextInfo, Page 458

Related Topics

CRPEJob::GetGraphText, Page 410

CRPEJob::GetGraphType

Use GetGraphType to retrieve which one of the available chart types is being used for the specified chart in the specified section. Use this method to find out what type of chart is being displayed in the report. There are many types of charts possible with Seagate Crystal Reports. This method will return the type of chart being displayed.

Syntax

```
GetGraphType (sectionCode SMALLINT, graphN SMALLINT) RETURNING BOOLEAN,
SMALLINT
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart from which you want to retrieve type information.
graphN	Specifies from which chart within the section you want to retrieve chart type information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

Chart Type Constants (page 467)

Related Topics

PEGetGraphTypeinfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetGroupCondition

Use GetGroupCondition to retrieve 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 437*, to change the group condition once it is known.

Syntax

```
GetGroupCondition (sectionCode SMALLINT) RETURNING BOOLEAN, ixString,
SMALLINT, SMALLINT
```

Parameter

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to retrieve group condition information.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

A ixString object containing the name of the group summary field.

A group condition field type constant which can be decoded using one of the following masks to separate the condition type from the group condition in the condition parameter.

- PEP_GC_CONDITIONMASK: Obtains the group condition value. Use the bitwise AND (&) to combine this mask with the value of the condition 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 condition parameter to obtain a value representing the type of field used by the group condition.

A **Sort Order Constants** (page 470).

Remarks

For group condition field types other than Date and Boolean, the group condition value of the *condition* parameter is PEP_GC_ANYCHANGE.

Value (Date Fields)	***Meaning***
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.
Value (Boolean Fields)	***Meaning***
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.

Value (Boolean Fields)	*Meaning*
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:

Value	*Meaning*
PEP_GC_TYPEOTHER	Any field type other than Date or Boolean. The group condition portion of the *condition* 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.

Related Topics

PEGetGroupCondition in the *Seagate Crystal Reports Technical Reference Guide*

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 439*, to identify and then change an existing group selection formula at print time in response to a user selection.

Return Value

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 in the *Seagate Crystal Reports Technical Reference Guide*

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::StartJob, Page 450* was called.

Return Values

<table>
<tr><td>SMALLINT</td><td colspan="2">Returns one of the following values according to the status of the current print job:</td></tr>
<tr><td></td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td>-1</td><td>The Crystal Report Engine has not been opened, or a print job has not been established.</td></tr>
<tr><td></td><td>PEP_JOBNOTSTARTED</td><td>The job has not been started.</td></tr>
<tr><td></td><td>PEP_JOBINPROGRESS</td><td>The job is currently in progress.</td></tr>
<tr><td></td><td>PEP_JOBCOMPLETED</td><td>The job has completed successfully.</td></tr>
<tr><td></td><td>PEP_JOBFAILED</td><td>The job has failed.</td></tr>
<tr><td></td><td>PEP_JOBCANCELLED</td><td>The job has been cancelled by the user.</td></tr>
<tr><td>CRPEJobInfo</td><td colspan="2">Class CRPEJobInfo.</td></tr>
</table>

Related Topics

PEGetJobStatus in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetLineHeight

Syntax

```
GetLineHeight (sectionCode SMALLINT, lineN SMALLINT) RETURNING BOOLEAN,
SMALLINT, SMALLINT
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to retrieve line height.
lineN	Specifies the 0-based line number within the specified section for which you want to retrieve the height. 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 440*.

Return Value

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 440*, to change report margins.

Return Value

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 in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetMinimumSectionHeight

Syntax

```
GetMinimumSectionHeight (sectionCode SMALLINT) RETURNING BOOLEAN, SMALLINT
```

Parameter

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to retrieve minimum section height.

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 441*.

Return Values

BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
SMALLINT	Retrieves the minimum height, in twips, for the specified report section.

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 441*.

Return Values

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 in the *Seagate Crystal Reports Technical Reference Guide*

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 419*.

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 in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNGroups

Use GetNGroups to retrieve the number of group sections in the report.

Syntax

```
GetNGroups () RETURNING SMALLINT
```

Returns

- Returns the number of group sections in the report.
- Returns -1 if an error occurs.

Related Topics

PEGetNGroups in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNGroupSortFields

Use GetNGroupSortFields to retrieve the number of group sort fields in the specified report. This method is typically used as one of a series: GetNGroupSortFields (called once); *CRPEJob::GetNthSortField, Page 421*, (called as many times as needed to identify the correct group sort field); and *CRPEJob::SetNthSortField, Page 443*, (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.

Syntax

```
GetNGroupSortFields () RETURNING SMALLINT
```

Returns

- Returns 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 in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNLinesInSection

Use this method to determine the number of lines in the specified section of the report, that is, how long a specified report section is according to the number of lines. For example, this is useful when trying to fit mailing label reports into the labels you are using.

Syntax

```
GetNLinesInSection (sectionCode SMALLINT) RETURNING SMALLINT
```

Parameter

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to retrieve the number of lines.

Returns

- Returns 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 421*, or *CRPEJob::SetNthParam, Page 443*.

Return Value

The number of parameters in the current stored procedure being used to generate the report.

Related Topics

PEGetNParameterFields in the *Seagate Crystal Reports Technical Reference Guide*

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 in the *Seagate Crystal Reports Technical Reference Guide*

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

formulaN	Specifies the 0-based 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 434*.

Return Values

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 in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthGroupSortField

Syntax

```
GetNthGroupSortField (sortFieldN SMALLINT) RETURNING BOOLEAN, ixString,
SMALLINT
```

Parameter

sortFieldN	Specifies the 0-based 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 417* (called once), CRPEJob::GetNthGroupSortField (called as many times as needed to identify the correct group sort field), and *CRPEJob::SetNthGroupSortField, Page 442* (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.

Returns

BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Retrieves the name of the group sort field.
SMALLINT	Retrieves one of the **Sort Order Constants** (page 470).

Related Topics

PEGetNthGroupSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthParam

Syntax

```
GetNthParam (paramN SMALLINT) RETURNING BOOLEAN, ixString
```

Parameter

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.

Returns

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

PEGetNthParameterCurrentValue in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthSortField

Syntax

```
GetNthSortField (sortFieldN SMALLINT) RETURNING BOOLEAN, ixString, SMALLINT
```

Parameter

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).

Returns

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	Retrieves one of the **Sort Order Constants** (page 470).

Related Topics

PEGetNthSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthTableLocation

Use GetNthTableLocation to retrieve the table location information for the specified table in the report. This method is typically combined with *CRPEJob::SetNthTableLocation, Page 444*, to identify the location of a table and then to change it.

Syntax

```
GetNthTableLocation (tableN SMALLINT) RETURNING BOOLEAN, CRPETableLocation
```

Parameter

tableN	Specifies the 0-based number of the table for which you want to retrieve location information.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPETableLocation (page 466)

Related Topics

PEGetNthTableLocation in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthTableLogOnInfo

Use GetNthTableLogOnInfo to retrieve SQL connection information for the specified table.

Syntax

```
GetNthTableLogonInfo (tableN SMALLINT) RETURNING BOOLEAN, CRPELogonInfo
```

Parameter

tableN	Specifies the 0-based number of the table from which you want to get log on information.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPELogOnInfo (page 461)

Related Topics

PEGetNthTableLogOnInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthTableSessionInfo

Use GetNthTableSessionInfo to retrieve the 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.

Syntax

```
GetNthTableSessionInfo (tableN SMALLINT) RETURNING BOOLEAN, CRPESessionInfo
```

Parameter

tableN	A 0-based table number indicating for which MS Access table in the report you want to obtain the session information.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPESessionInfo (page 465)

Related Topics

PEGetNthTableSessionInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetNthTableType

Use GetNthTableType to retrieve the table type information for the specified table.

Syntax

```
GetNthTableType (tableN SMALLINT) RETURNING BOOLEAN, CRPETableType
```

Parameter

tableN	The 0-based table number indicating for which table in the report you want to determine the type.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPETableType (page 466)

Related Topics

PEGetNthTableType in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetPrintDate

Use GetPrintDate to retrieve the print date (if any) that was specified with the report. Use this method to retrieve the print date and pass it back using *CRPEJob::SetPrintDate, Page 446*.

Syntax

```
GetPrintDate () RETURNING BOOLEAN, SMALLINT, SMALLINT, SMALLINT
```

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

The year component of the current print date.

The month component of the current print date.

The day component of the current print date.

Related Topics

PEGetPrintDate in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetPrintOptions

Use GetPrintOptions to retrieve the print options specified for the report (the options that are set in the Print Setup common dialog box) and use them to fill in the **class CRPEPrintOptions** (page 462). Use this function to retrieve print options from the report in order to update them and pass back using *CRPEJob::SetPrintOptions, Page 446*.

Syntax

```
GetPrintOptions () RETURNING BOOLEAN, CRPEPrintOptions
```

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPEPrintOptions (page 462).

Related Topics

PEGetPrintOptions in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetReportTitle

Use GetReportTitle to retrieve the report title for the print job.

Syntax

```
GetReportTitle () RETURNING BOOLEAN, ixString
```

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

A ixString object containing the report title.

Related Topics

PEGetReportTitle in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::GetSectionFormat

Use GetSectionFormat to retrieve the section format settings for a selected section in the specified report and supply the information by assigning values to the members of **class CRPESectionOptions** (page 463). Use this method in order to edit and update the section formats and pass the information back using *CRPEJob::SetSectionFormat, Page 447*.

Syntax

```
GetSectionFormat (sectionCode SMALLINT) RETURNING BOOLEAN, CRPESectionOptions
```

Parameter

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to retrieve format information.

Returns

Boolean

- TRUE if the call is successful.
- FALSE if the call fails.

class CRPESectionOptions (page 463).

Related Topics

PEGetSectionFormat in the *Seagate Crystal Reports Technical Reference Guide*

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.

Returns

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 DEVMODE in the *Seagate Crystal Reports Technical Reference Guide*.

Related Topics

PEGetSelectedPrinter in the *Seagate Crystal Reports Technical Reference Guide*

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 427*, to identify and then change an existing selection formula at print time in response to a user selection.

Returns

BOOLEAN	TRUE if the call is successful, FALSE if something goes wrong.
ixString	Specifies the existing selection formula for the report.

Related Topics

PEGetSelectionFormula in the *Seagate Crystal Reports Technical Reference Guide*

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, Page 449* to retrieve and then change the SQL query for the report.

Return Values

TRUE if the call is successful, FALSE if something goes wrong.

ixString	Assigned the text of the SQL query being sent to the server if the call completes successfully.

Related Topics

PEGetSQLQuery in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::IsJobFinished

Syntax

```
IsJobFinished () RETURNING BOOLELAN
```

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 in the *Seagate Crystal Reports Technical Reference Guide*

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 in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::OutputToPrinter

Use OutputToPrinter to set 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::StartJob, Page 450*.

Syntax

```
OutputToPrinter (nCopies SMALLINT : 1) RETURNING BOOLEAN
```

Parameter

nCopies	Specifies how many copies of the report are to be printed. Default is 1 copy.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEOutputToPrinter in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::OutputToWindow

Use OutputToWindow to set the output of the print job to the preview window which will have the specified attributes. This method does not print the report to the window, but specifies that when the report is printed, it will appear in a preview window. To actually print the report, use *CRPEJob::StartJob, Page 450*.

Syntax

```
OutputToWindow (title CHAR (*), left SMALLINT, top SMALLINT, width SMALLINT,
height SMALLINT, style INTEGER, parentWindow ixWindow) RETURNING BOOLEAN
```

Parameters

title	Specifies the title that 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEOutputToWindow in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::PrintControlsShowing

Syntax

```
PrintControlsShowing () RETURNING BOOLEAN, SMALLINT
```

Remarks

Checks if the print controls are displayed in the preview window. Use *CRPEJob::ShowPrintControls, Page 450*, to change whether or not print controls will appear in the preview window.

Returns

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 in the *Seagate Crystal Reports Technical Reference Guide*

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.).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEPrintWindow in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SelectPrinter

Syntax

```
SelectPrinter (driverName CHAR (*), printerName CHAR (*), portName CHAR (*),
mode FOREIGN) RETURNING BOOLEAN
```

Parameters

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 DEVMODE in the *Seagate Crystal Reports Technical Reference Guide*.

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetPrintDate in the *Seagate Crystal Reports Technical Reference Guide*

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

<table>
<tr><td>sectionCode</td><td colspan="2">Specifies the section of the report for which you want to set the font. Use one of the following values:</td></tr>
<tr><td></td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td>PEP_ALLSECTIONS</td><td>Sets the line height for all sections.</td></tr>
<tr><td></td><td>PEP_HEADERSECTION</td><td>Sets the line height for the Page Header section.</td></tr>
<tr><td></td><td>PEP_GROUPHEADER</td><td>Sets the line height for the Group Header section.</td></tr>
<tr><td></td><td>PEP_DETAILSECTION</td><td>Sets the line height for the Details section.</td></tr>
<tr><td></td><td>PEP_GROUPFOOTER</td><td>Sets the line height for the Group Footer section.</td></tr>
<tr><td></td><td>PEP_GRANDTOTALSECTION</td><td>Sets the line height for the Grand Total section.</td></tr>
<tr><td></td><td>PEP_FOOTERSECTION</td><td>Sets the line height for the Page Footer section.</td></tr>
<tr><td>scopeCode</td><td colspan="2">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:</td></tr>
<tr><td></td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td>PEP_FIELDS</td><td>Sets the default font for fields in the report section specified.</td></tr>
<tr><td></td><td>PEP_TEXT</td><td>Sets the default font for all text (that has not been entered as a text field value) in the report section specified.</td></tr>
<tr><td>faceName</td><td colspan="2">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”.</td></tr>
<tr><td>fontFamily</td><td colspan="2">Specifies the font family for the font you want to use. Use one of the following values:</td></tr>
<tr><td></td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td>FF_DONTCARE</td><td>No font family or family does not matter.</td></tr>
<tr><td></td><td>FF_ROMAN</td><td>Variable pitch font with serifs.</td></tr>
<tr><td></td><td>FF_SWISS</td><td>Fixed pitch font without serifs.</td></tr>
</table>

	Value	***Meaning***
	FF_MODERN	Fixed pitch font, with or without serifs.
	FF_SCRIPT	Handwriting-like font.
	FF_DECORATIVE	Fancy display font.
fontPitch	Specifies the font pitch you wish to use. Use one of the following values:	
	Value	***Meaning***
	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.
charSet	Specifies the character set you wish to use. Use one of the following values:	
	Value	
	ANSI_CHARSET	
	DEFAULT_CHARSET	
	SYMBOL_CHARSET	
	HANGEFUL_CHARSET	
	OEM_CHARSET	
	SHIFTJIS_CHARSET	
	CHINESEBIG5_CHARSET	
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:	
	Value	
	FW_DONTCARE	

Value
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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetFont in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetFormula

Syntax

```
SetFormula (formulaName CHAR(*), formulaText CHAR(*)) RETURNING BOOLEAN
```

Parameters

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 408*, to identify and then change an existing formula at print time in response to a user selection.

Returns

- TRUE if the call is successful.
- FALSE there is an error in the formula, or if the call fails.

Related Topics

PESetFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGraphData

Use SetGraphData to change the data that is used from your report to create a specified chart. This information includes the groups used to create the rows and columns of the chart and the summary field used to set the values of the risers in the chart.

Syntax

```
SetGraphData (sectionCode SMALLINT, graphN SMALLINT, graphDataInfo
CRPEGraphDataInfo) RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart for which you want to set data information.
graphN	Specifies from which chart within the section you want to set chart data information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.
graphDataInfo	Specifies a pointer to **class CRPEGraphDataInfo** (page 456).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetGraphTextInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGraphOptions

Use SetGraphOptions to change any of several chart options. These options include the minimum and maximum values that can appear on the chart, whether grid lines appear, whether risers are labeled, whether bar charts have horizontal or vertical bars, and whether a legend appears on the chart.

Syntax

```
SetGraphOptions (sectionCode SMALLINT, graphN SMALLINT, graphOptions
CRPEGraphOptions) RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart for which you want to set options information.
graphN	Specifies from which chart within the section you want to set chart options information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.
graphOptions	Specifies a pointer to **class CRPEGraphOptions** (page 457).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetGraphOptionInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGraphText

Use SetGraphText to set or change the text that appears with a chart. A chart 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 chart).

Syntax

```
SetGraphText (sectionCode SMALLINT, graphN SMALLINT, graphTextInfo
CRPEGraphTextInfo) RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart for which you want to set text information.
graphN	Specifies from which chart within the section you want to set chart text information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.

graphTextInfo	Specifies a pointer to **class CRPEGraphTextInfo** (page 458).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetGraphTextInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGraphType

Use SetGraphType to set the type of chart displayed for the specified chart in the specified section based on one of the available types. Use this method to change the type of chart that is displayed in a report.

Syntax

```
SetGraphType (sectionCode SMALLINT, graphN SMALLINT, graphType SMALLINT)
RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report containing the chart for which you want to set type information.
graphN	Specifies from which chart within the section you want to set chart type information. This value is 0-based. Within a section, charts are numbered from top to bottom first and from left to right if they have the same top.
graphType	Specifies one of the **Chart Type Constants** (page 467), indicating the style of the chart that you want to set.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetGraphTypeinfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGroupCondition

Syntax

```
SetGroupCondition (sectionCode SMALLINT,conditionField CHAR(*), condition
SMALLINT, sortDirection SMALLINT) RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to set group condition information.
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.

Value (Date Fields)	***Meaning***
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.

Value (Boolean Fields)	***Meaning***
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.

	Value (Boolean Fields)	*Meaning*
	PEP_GC_NEXTISNO	Triggers a grouping every time the next value in the sort and group by field is No.
sortDirection	Specifies one of **Sort Order Constants** (page 470).	

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 cannot 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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetGroupCondition in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetGroupSelectionFormula

Use SetGroupSelectionFormula to set the formula text for the group selection formula. This method is often used with *CRPEJob::GetGroupSelectionFormula, Page 413* to identify and then change an existing group selection formula in response to a user selection at print time.

Syntax

```
SetGroupSelectionFormula (formulaText CHAR(*)) RETURNING BOOLEAN
```

Parameter

formulaText	Specifies the new group selection formula that you want to assign to the report.

Returns

- TRUE if the call is successful.
- FALSE if there is an error in the formula, or if the call fails.

Related Topics

PESetGroupSelectionFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetLineHeight

Use SetLineHeight to set 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 in relation to the font size. For example, this method might be used to specify a 12 point line with 10 point type.

Syntax

```
SetLineHeight (sectionCode SMALLINT, lineN SMALLINT, height SMALLINT, ascent
SMALLINT) RETURNING BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to set line height.
lineN	Specifies the 0-based number of 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.

Returns

- TRUE if the call is successful.
- FALSE if there is an invalid selection, or if the call fails.

CRPEJob::SetMargins

Use SetMargins to set the page margins for the print job. Use this method any time you want to allow the user to change margins.

Syntax

```
SetMargins (left SMALLINT, right SMALLINT, top SMALLINT, bottom SMALLINT)
RETURNING BOOLEAN
```

Parameters

For any of the above parameters, PM_SM_DEFAULT can be used to specify that the report use default printer margins.

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetMargins in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetMinimumSectionHeight

Use SetMinimumSectionHeight to set 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.

Syntax

```
SetMinimumSectionHeight (sectionCode SMALLINT, height SMALLINT) RETURNING
BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to set minimum section height.
height	Specifies the minimum height in twips of the specified section.

Returns

- TRUE if the call is successful.
- FALSE if an invalid section has been specified, or if the call fails.

CRPEJob::SetNDetailCopies

Use SetNDetailCopies to set 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.

Syntax

```
SetNDetailCopies (nCopies SMALLINT) RETURNING BOOLEAN
```

Parameter

nCopies	Specifies the number of copies of the detail section of the report to be printed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNDetailCopies in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetNthGroupSortField

Use SetNthGroupSortField to specify that the specified group summary field is sorted in the specified direction (ascending or descending). This method does not create a new group, but will sort an existing group summary field.

Syntax

```
SetNthGroupSortField (sortFieldN SMALLINT, field CHAR(*), direction SMALLINT)
RETURNING BOOLEAN
```

Parameters

sortFieldN	Specifies the 0-based number of the group sort field that 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 one of the **Sort Order Constants** (page 470).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthGroupSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetNthParam

Syntax

```
SetNthParam (paramN SMALLINT, paramValue CHAR (*)) RETURNING BOOLEAN
```

Parameters

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".

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthParameterField in the *Seagate Crystal Reports Technical Reference Guide2*

CRPEJob::SetNthSortField

Use SetNthSortField to specify that the report data is sorted according to the specified field and direction.

Syntax

```
SetNthSortField (sortFieldN SMALLINT, field CHAR(*), direction SMALLINT)
RETURNING BOOLEAN
```

Parameters

sortFieldN	Specifies the 0-based number of the sort field that 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 one of the **Sort Order Constants** (page 470).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthSortField in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetNthTableLocation

Use SetNthTableLocation to set the table location information for the specified table in the report. This method is typically combined with *CRPEJob::GetNthTableLocation, Page 422,* to identify the location of a table and then to change it.

Syntax

```
SetNthTableLocation (tableN SMALLINT, tableLocation CRPETableLocation)
RETURNING BOOLEAN
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to set a new location.
tableLocation	Specifies a pointer to **class CRPETableLocation** (page 466).

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthTableLocation in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetNthTableLogOnInfo

Syntax

```
SetNthTableLogonInfo (tableN SMALLINT, logonInfo CRPELogOnInfo, propagate
BOOLEAN) RETURNING BOOLEAN
```

Parameters

tableN	Specifies the 0-based number of the table for which you want to set log on information.
logonInfo	Specifies a pointer to **class CRPELogOnInfo** (page 461).

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthTableLogOnInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetNthTableSessionInfo

Syntax

```
SetNthTableSessionInfo (tableN SMALLINT, sessionInfo CRPESessionInfo,
propagate BOOLEAN) RETURNING BOOLEAN
```

Parameters

tableN	0 indexed table number indicating which MS Access table in the report the session is being opened for.
sessionInfo	Specifies a pointer to **class CRPESessionInfo** (page 465).
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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetNthTableSessionInfo in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetPrintDate

Syntax

```
SetPrintDate (v_year SMALLINT, v_month SMALLINT, v_day SMALLINT) RETURNING
BOOLEAN
```

Parameters

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetPrintDate in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetPrintOptions

Syntax

```
SetPrintOptions (options CRPEPrintOptions) RETURNING BOOLEAN
```

Parameter

options	Specifies a pointer to **class CRPEPrintOptions** (page 462).

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetPrintOptions in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetReportTitle

Syntax

```
SetReportTitle (title CHAR(*)) RETURNING BOOLEAN
```

Parameter

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetReportTitle in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetSectionFormat

Syntax

```
SetSectionFormat (sectionCode SMALLINT, options CRPESectionOptions) RETURNING
BOOLEAN
```

Parameters

sectionCode	Specifies the **Section Codes** (page 470), for the section of the report for which you want to set the section format.
options	Specifies a pointer to **class CRPESectionOptions** (page 463).

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 463*.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetSectionFormat in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetSelectionFormula

Syntax

```
SetSelectionFormula (formulaText CHAR(*)) RETURNING BOOLEAN
```

Parameter

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 427*, to identify and then change an existing selection formula at print time in response to a user selection.

Returns

- TRUE if the call is successful.
- FALSE if there is an error in the formula, or if the call fails.

Related Topics

PESetSelectionFormula in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::SetSQLQuery

Syntax

```
SetSQLQuery (query CHAR(*)) RETURNING BOOLEAN
```

Parameter

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 427* to retrieve and then change the SQL query for the report.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PESetSQLQuery in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::Show...Page

Use these methods to show the specified page in the preview window. Use these methods to customize how a user moves through the pages of a report.

Syntax

```
ShowFirstPage () RETURNING BOOLEAN
ShowLastPage () RETURNING BOOLEAN
ShowNextPage () RETUNING BOOLEAN
ShowPreviousPage () RETURNING BOOLEAN
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEShow...Page in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::ShowPrintControls

Use ShowPrintControls to toggle the display of the preview window control buttons. Use this to display default controls, or to hide default controls and provide customized controls.

Syntax

```
ShowPrintControls (showControls BOOLEAN) RETURNING BOOLEAN
```

Parameter

showControls	Boolean value indicating whether print controls will be showing in the preview window. TRUE indicates default print controls are to be displayed in the preview window. FALSE indicates no controls are to be displayed.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEShow...Page in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::StartJob

Use StartJob 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.

Syntax

```
StartJob () RETURNING BOOLEAN
```

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEStartPrintJob in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::TestNthTableConnectivity

Use TestNthTableConnectivity to determine 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.

Syntax

```
TestNthTableConnectivity (tableN SMALLINT) RETURNING BOOLEAN
```

Parameter

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.

Returns

- TRUE if the database session, log on, and location information is all correct.
- FALSE if one or more of the above are incorrect, or if the call fails.

Related Topics

PETestNthTableConnectivity in the *Seagate Crystal Reports Technical Reference Guide*

CRPEJob::ZoomPreviewWindow

Syntax

```
ZoomPreviewWindow (level SMALLINT) RETURNING BOOLEAN
```

Parameter

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.

Returns

- TRUE if the call is successful.
- FALSE if the call fails.

Related Topics

PEZoomPreviewWindow in the *Seagate Crystal Reports Technical Reference Guide*

OTHER NEWERA CLASSES

The following classes are discussed in this section.

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 407*, and *CRPEJob::ExportTo, Page 406*.

Data Members

m_formatType	Specifies the export format to be used.
m_formatDLLName	Specifies the name of the format DLL that contains the export format to be used.
m_formatOptions	Provides additional export information specific to the format type being used.
m_nFormatOptionsBytes	Automatically assigned by CRPEJob::GetExportOptions. Unused by CRPEJob::SetExportOptions.
m_destinationType	Specifies the destination type of the exported report.
m_destinationDLLName	Specifies the name of the destination DLL that contains the destination type to be used.

m_destinationOptions	Provides additional information specific to the export destination type.
m_nDestinationOptionsBytes	Automatically assigned by CRPEJob::GetExportOptions. Unused by CRPEJob::SetExportOptions.

Constructor CRPEExportOptions::CRPEExportOptions

CRPEExportOptions::CRPEExportOptions 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.

Constructor Default Syntax

```
CRPEExportOptions ();
```

Constructor Syntax

```
CRPEExportOptions (formatDLLName CHAR(*) : NULL, formatType INTEGER: 0,
formatOptions FOREIGN : NULL, destinationDLLName CHAR(*) : NULL,
destinationType INTEGER : 0, destinationOptions FOREIGN : NULL);
```

Constructor Parameters

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:

To export a report in this format:	*Use this DLL:*
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

	To export a report in this format:	***Use this DLL:***
	Lotus 1-2-3 (WK3)	uxfwks.dll
	Excel 2.1	uxfxls.dll
	Excel 3.0	uxfxls.dll
	Excel 4.0	uxfxls.dll
formatType	Specifies the export format to be used. Assigns this value to the CRPEExportOptions::m_formatType member. Select from the following values:	
	To export a report in this format:	***Use this for formatType:***
	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
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:	
	To export a report in this format:	***Use this class:***
	Data Interchange Format	UXFDIFOptions

	To export a report in this format:	***Use this class:***
	Record Style Format (columns)	UXFRecordStyleOptions
	Comma Separated Values (CSV)	UXFCommaTabSeparated-Options
	Tab Separated Values	UXFCommaTabSeparated-Options
	Character Separated Values	UXFCharSeparatedOptions
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:	
	To export a report to this destination:	***Use this DLL name:***
	Disk File	uxddisk.dll
	E-mail (MAPI)	uxdmapi.dll
	E-mail (VIM)	uxdvim.dll
destinationType	Specifies the destination type of the exported report. Assigns this value to the CRPEExportOptions::m_destinationType member. The following values are available:	
	To export a report to this destination:	***Use this destination type:***
	Disk File	UXDDiskType
	E-mail (MAPI)	UXDMapiType
	E-mail (VIM)	UXDVIMType
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:	
	To export a report to this destination:	***Use this class:***
	Disk File	UXDDiskOptions
	E-mail (MAPI)	UXDMAPIOptions
	E-mail (VIM)	UXDVIMOptions

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 408*, to retrieve information regarding an existing graph and by *CRPEJob::SetGraphData, Page 435*, to change the data used by an existing graph.

Data Members

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.

Constructor CRPEGraphDataInfo::CRPEGraphDataInfo

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.

Constructor Default Syntax

```
CRPEGraphDataInfo ();
```

Constructor Syntax

```
CRPEGraphDataInfo (rowGroupN SMALLINT : 0, colGroupN SMALLINT: 0
summarizedFieldN SMALLINT : 0, graphDirection SMALLINT : PEP_GRAPH_COLS_ONLY);
```

Constructor Parameters

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*	*Description*
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.

class CRPEGraphOptions

The CRPEGraphOptions class contains information on several options available with graphs and charts. This class is used by *CRPEJob::GetGraphOptions, Page 409*, to determine what options have been set for a chart, and it is used by *CRPEJob::SetGraphOptions, Page 436*, to change the options for a chart.

Data Members

m_graphMaxValue	Specifies the maximum value that will appear in the graph. Any graph values above this value are not charted.
m_graphMinValue	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.

Constructor CRPEGraphOptions::CRPEGraphOptions

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.

Constructor Default Syntax

```
CRPEGraphOptions()
```

Constructor Syntax

```
CRPEGraphOptions (graphMaxValue FLOAT : 0, graphMinValue FLOAT : 0,
showDataValue BOOLEAN : FALSE, showGridLine BOOLEAN : FALSE, verticalBars
BOOLEAN : FALSE, showLegend BOOLEAN : FALSE, fontFaceName CHAR (*) : NULL)
```

Constructor Parameters

graphMaxValue	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_graphMaxValue member variable.
graphMinValue	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_graphMinValue member variable.
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.

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 410*,and *CRPEJob::SetGraphText, Page 436*.

Data Members

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.

Constructor CRPEGraphTextInfo::CRPEGraphTextInfo

Constructor Default Syntax

```
CRPEGraphTextInfo ();
```

Constructor Syntax

```
CRPEGraphTextInfo (graphSubTitle CHAR(*) : NULL, graphFootNote CHAR(*) : NULL,
graphGroupsTitle CHAR(*) : NULL, graphSeriesTitle CHAR(*) : NULL,
graphXAxisTitle CHAR(*) : NULL, graphYAxisTitle CHAR(*) : NULL,
graphZAxisTitle CHAR(*) : NULL);
```

Constructor Parameters

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 414*. Status information concerning the job is returned through this class.

Data Members

Upon construction of the structure, all members are initialized to zero except m_printEnded which is initialized to FALSE.

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.

Constructor CRPEJobInfo::CRPEJobInfo

Constructs a CRPEJobInfo class object. Initializes all required members of the class.

Constructor Syntax

```
CRPEJobInfo (numRecordsRead INTEGER : 0, numRecordsSelected INTEGER : 0,
numRecordsPrinted INTEGER : 0, displayPageN SMALLINT : 0, latest PageN
SMALLINT : 0, startPageN SMALLINT : 0, printEnded BOOLEAN : FALSE);
```

class CRPELogOnInfo

This class is used by member functions *CRPEngine::LogOnServer, Page 396*, *CRPEngine::LogOffServer, Page 395*, *CRPEJob::GetNthTableLogOnInfo, Page 423*, *CRPEJob::SetNthTableLogOnInfo, Page 444*. The class is used to contain SQL connection information.

Data Members

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.

Constructor CRPELogOnInfo::CRPELogOnInfo

Constructor Default Syntax

```
CRPELogOnInfo ();
```

Constructor Syntax

```
CRPELogOnInfo (serverName CHAR (*), databaseName CHAR (*), userID CHAR (*),
password CHAR (*));
```

Constructor Parameters

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.
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 446*.

Data Members

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.

Constructor CRPEPrintOptions::CRPEPrintOptions

Constructor Default Syntax

```
CRPEPrintOptions ();
```

Constructor Syntax

```
CRPEPrintOptions (startPageN SMALLINT : 0, stopPageN SMALLINT : 0,
nReportCopies SMALLINT : 0, collation SMALLINT : PEP_DEFAULTCOLLATION);
```

Constructor Parameters

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 426*, and *CRPEJob::SetSectionFormat, Page 447*.

Data Members

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.

Constructor CRPESectionOptions::CRPESectionOptions

Constructor Syntax

```
CRPESectionOptions (visible BOOLEAN : FALSE, newPageBefore BOOLEAN : FALSE,
newPageAfter BOOLEAN : FALSE, keepTogether BOOLEAN : FALSE, suppressBlankLines
BOOLEAN : FALSE, resetPageNAfter BOOLEAN : FALSE, printAtBottomOfPage BOOLEAN
: FALSE)
```

Constructor Parameters

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 423*,and *CRPEJob::SetNthTableSessionInfo, Page 445*.

Data Members

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.

Constructor CRPESessionInfo::CRPESessionInfo

Constructor Default Syntax

```
CRPESessionInfo ();
```

Constructor Syntax

```
CRPESessionInfo (userID CHAR(*) : NULL, password CHAR(*) : NULL, sessionHandle
INTEGER : 0);
```

Constructor Parameters

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 422,* and *CRPEJob::SetNthTableLocation, Page 444.*

Data Member

m_location	Specifies the database location.

Constructor CRPETableLocation::CRPETableLocation

Constructor Syntax

```
CRPETableLocation (location CHAR(*) : NULL);
```

Constructor Parameter

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 423.*

Data Members

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.

Constructor CRPETableType::CRPETableType

Constructs a CRPETableType class object. Initializes all members of the class.

Constructor Syntax

```
CRPETableType (dbType SMALLINT : PEP_DT_SQL, dllName CHAR(*) : NULL,
descriptiveName: CHAR(*) : NULL);
```

CLASS CONSTANTS

The following constants, some of the more commonly used class constants, are discussed in this section.

Chart Type Constants

Constant	*Description*
PEP_SIDE_BY_SIDE_BAR_GRAPH	Side By Side bar charts.
PEP_STACKED_BAR_GRAPH	Stacked bar charts.
PEP_PERCENT_BAR_GRAPH	Percent bar charts.
PEP_FAKED_3D_SIDE_BY_SIDE_BAR_GRAPH	3D Side By Side bar charts.
PEP_FAKED_3D_STACKED_BAR_GRAPH	3D Stacked bar charts.
PEP_FAKED_3D_PERCENT_BAR_GRAPH	3D Percent bar charts.
PEP_PIE_GRAPH	Pie charts.
PEP_MULTIPLE_PIE_GRAPH	Multiple Pie charts.
PEP_PROPORTIONAL_MULTI_PIE_GRAPH	Weighted Pie charts.
PEP_LINE_GRAPH	Line charts.
PEP_AREA_GRAPH	Area charts.
PEP_THREED_BAR_GRAPH	3D bar charts.
PEP_USER_DEFINED_GRAPH	User Defined chart type.
PEP_UNKNOWN_TYPE_GRAPH	Unknown chart type.

Error Codes

Constant	*Value*	*Description*
CONSTANT PEP_ERR_NOERROR	INTEGER = 0	
CONSTANT PEP_ERR_NOTENOUGHMEMORY	INTEGER = 500	
CONSTANT PEP_ERR_INVALIDJOBNO	INTEGER = 501	
CONSTANT PEP_ERR_INVALIDHANDLE	INTEGER = 502	
CONSTANT PEP_ERR_STRINGTOOLONG	INTEGER = 503	
CONSTANT PEP_ERR_NOSUCHREPORT	INTEGER = 504	
CONSTANT PEP_ERR_NODESTINATION	INTEGER = 505	
CONSTANT PEP_ERR_BADFILENUMBER	INTEGER = 506	
CONSTANT PEP_ERR_BADFILENAME	INTEGER = 507	
CONSTANT PEP_ERR_BADFIELDNUMBER	INTEGER = 508	
CONSTANT PEP_ERR_BADFIELDNAME	INTEGER = 509	
CONSTANT PEP_ERR_BADFORMULANAME	INTEGER = 510	
CONSTANT PEP_ERR_BADSORTDIRECTION	INTEGER = 511	
CONSTANT PEP_ERR_ENGINENOTOPEN	INTEGER = 512	
CONSTANT PEP_ERR_INVALIDPRINTER	INTEGER = 513	
CONSTANT PEP_ERR_PRINTFILEEXISTS	INTEGER = 514	
CONSTANT PEP_ERR_BADFORMULATEXT	INTEGER = 515	
CONSTANT PEP_ERR_BADGROUPSECTION	INTEGER = 516	
CONSTANT PEP_ERR_ENGINEBUSY	INTEGER = 517	
CONSTANT PEP_ERR_BADSECTION	INTEGER = 518	
CONSTANT PEP_ERR_NOPRINTWINDOW	INTEGER = 519	
CONSTANT PEP_ERR_JOBALREADYSTARTED	INTEGER = 520	
CONSTANT PEP_ERR_BADSUMMARYFIELD	INTEGER = 521	
CONSTANT PEP_ERR_NOTENOUGHSYSRES	INTEGER = 522	
CONSTANT PEP_ERR_BADGROUPCONDITION	INTEGER = 523	
CONSTANT PEP_ERR_JOBBUSY	INTEGER = 524	
CONSTANT PEP_ERR_BADREPORTFILE	INTEGER = 525	
CONSTANT PEP_ERR_NODEFAULTPRINTER	INTEGER = 526	
CONSTANT PEP_ERR_SQLSERVERERROR	INTEGER = 527	
CONSTANT PEP_ERR_BADLINENUMBER	INTEGER = 528	
CONSTANT PEP_ERR_DISKFULL	INTEGER = 529	

Constant	*Value*	*Description*
CONSTANT PEP_ERR_FILEERROR	INTEGER = 530	
CONSTANT PEP_ERR_INCORRECTPASSWORD	INTEGER = 531	
CONSTANT PEP_ERR_BADDATABASEDLL	INTEGER = 532	
CONSTANT PEP_ERR_BADDATABASEFILE	INTEGER = 533	
CONSTANT PEP_ERR_ERRORINDATABASEDLL	INTEGER = 534	
CONSTANT PEP_ERR_DATABASESESSION	INTEGER = 535	
CONSTANT PEP_ERR_DATABASELOGON	INTEGER = 536	
CONSTANT PEP_ERR_DATABASELOCATION	INTEGER = 537	
CONSTANT PEP_ERR_BADSTRUCTSIZE	INTEGER = 538	
CONSTANT PEP_ERR_BADDATE	INTEGER = 539	
CONSTANT PEP_ERR_BADEXPORTDLL	INTEGER = 540	
CONSTANT PEP_ERR_ERRORINEXPORTDLL	INTEGER = 541	
CONSTANT PEP_ERR_PREVATFIRSTPAGE	INTEGER = 542	
CONSTANT PEP_ERR_NEXTATLASTPAGE	INTEGER = 543	
CONSTANT PEP_ERR_CANNOTACCESSREPORT	INTEGER = 544	
CONSTANT PEP_ERR_USERCANCELLED	INTEGER = 545	
CONSTANT PEP_ERR_OLE2NOTLOADED	INTEGER = 546	
CONSTANT PEP_ERR_BADCROSSTABGROUP	INTEGER = 547	
CONSTANT PEP_ERR_NOCTSUMMARIZEDFIELD	INTEGER = 548	
CONSTANT PEP_ERR_DESTINATIONNOTEXPORT	INTEGER = 549	
CONSTANT PEP_ERR_INVALIDPAGENUMBER	INTEGER = 550	
CONSTANT PEP_ERR_NOTSTOREDPROCEDURE	INTEGER = 552	
CONSTANT PEP_ERR_INVALIDPARAMETER	INTEGER = 553	
CONSTANT PEP_ERR_GRAPHNOTFOUND	INTEGER = 554	
CONSTANT PEP_ERR_INVALIDGRAPHTYPE	INTEGER = 555	
CONSTANT PEP_ERR_INVALIDGRAPHDATA	INTEGER = 556	
CONSTANT PEP_ERR_CANNOTMOVEGRAPH	INTEGER = 557	
CONSTANT PEP_ERR_INVALIDGRAPHTEXT	INTEGER = 558	
CONSTANT PEP_ERR_INVALIDGRAPHOPT	INTEGER = 559	
CONSTANT PEP_ERR_BADSECTIONHEIGHT	INTEGER = 560	
CONSTANT PEP_ERR_BADVALUETYPE	INTEGER = 561	
CONSTANT PEP_ERR_INVALIDSUBREPORTNAME	INTEGER = 562	
CONSTANT PEP_ERR_NOPARENTWINDOW	INTEGER = 564	No dialog parent window.

Constant	*Value*	*Description*
CONSTANT PEP_ERR_INVALIDZOOMFACTOR	INTEGER = 565	Invalid zoom factor.
CONSTANT PEP_ERR_PAGESIZEOVERFLOW	INTEGER = 567	
CONSTANT PEP_ERR_LOWSYSTEMRESOURCES	INTEGER = 568	
CONSTANT PEP_ERR_BADGROUPNUMBER	INTEGER = 570	
CONSTANT PEP_ERR_INVALIDNEGATIVEVALUE	INTEGER = 572	
CONSTANT PEP_ERR_INVALIDMEMORYPOINTER	INTEGER = 573	
CONSTANT PEP_ERR_INVALIDPARAMETERNUMBER	INTEGER = 594	
CONSTANT PEP_ERR_SQLSERVERNOTOPENED	INTEGER = 599	
CONSTANT PEP_ERR_NOTIMPLEMENTED	INTEGER = 999	

Section Codes

Constant	*Description*
PEP_ALLSECTIONS	All sections.
PEP_HEADERSECTION	Page Header section.
PEP_GROUPHEADER	Group Header section.
PEP_DETAILSECTION	Details section.
PEP_GROUPFOOTER	Group Footer section.
PEP_GRANDTOTALSECTION	Grand Total section.
PEP_FOOTERSECTION	Page Footer section.

Sort Order Constants

Constant	*Description*
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).

I N D E X

A

B

C

D

E

F

G

L

M

N

O

P

R

S

T

U

V

W

Z