crystal decisions
A SEAGATE COMPANY

Crystal Reports

Accessing Subreports from a Visual Basic application using API Calls

---

Overview

Since Seagate Crystal Reports 5.0 developers have been able to access subreport objects contained in reports using API calls. This document will demonstrate how to retrieve the subreport name, get the handle and open each subreport.

Once this is achieved then the subreport can be treated the same as a regular report. This sample will print the report, which uses a PC type database, to the preview window.

Contents

Getting Started

The first thing that must be done is to include into the project the module GLOBAL32.BAS (or GLOBAL.BAS for 16 bit users). The GLOBAL.BAS and GLOBAL32.BAS are modules that have declared structures for the Crystal Reports Print Engine. These structures are made available to VB that will allow us to make calls to the Print Engine. For this simple example, we will place all the code within three command buttons. Before you proceed, place three command buttons on your form.

There are only five print engine calls specifically associated with subreports. The purpose of these print engine calls is to retrieve the name of the subreport. Once the name of the subreport has been retrieved, the subreport can be opened and treated the same as a regular report. A subreport has the same properties and functionality as a main report, except that it is embedded within the main report and treated as a separate object.

The five print engine calls specifically associated with subreports are:

PEGetNSubreportsInSection - used to find the number of subreports in a section

PEGetNthSubreportInSection - used to retrieve the handle of the nth subreport in a section

PEGetSubreportInfo - used to retrieve the name of the subreport once the handle is retrieved

PEOpenSubreport - used to open a subreport

PECloseSubreport - used to close a subreport

Using these five print engine calls together with a few section management functions (PEGetNSection, PEGetSectionCode), one can write generic code that is able to find and open any subreport in a main report. Here is some sample code from a Visual Basic application that demonstrates this. This sample will preview the report, including the subreport, to the window.

Basic Code to Access Subreports

Within the **General Declarations** portion of the module, declare the following variables.

General Declarations

```
Dim I as Integer

Dim J as Integer

Dim Result as Integer

Dim MainJob as Integer
```

```
        Dim SubJob as Integer
        Dim NumSections as Integer
        Dim SectionCode as Integer
        Dim NumSubreports as Integer
        Dim SubreportHandle as Integer
        Dim MySubreportInfo as PESubreportInfo

Private Sub Command1_Click()
'Initialize subreport info structure size
MySubreportInfo.StructSize = PE_SIZEOF_SUBREPORT_INFO
'Open the print engine
Result = PEOpenEngine()
'Open the main report
MainJob = PEOpenPrintJob("<path to report file, including the .RPT file>")
'Get the number of sections in the main report
NumSections = PEGetNSections(MainJob)
'Loop through each section in the main report
  For I = 0 To (NumSections - 1)
    'Get the section code for the current section
        sectionCode = PEGetSectionCode(MainJob, I)
    'Get the number of subreports in the current section
        NumSubreports = PEGetNSubreportsInSection(MainJob,
sectionCode)
    'Loop through each subreport in the current section
        For J = 0 To (NumSubreports - 1)
'Get the handle to the current subreport in the current section
                subreportHandle =
PEGetNthSubreportInSection(MainJob, sectionCode, J)
```

```
'Get the name of the current subreportin the current section

                Result = PEGetSubreportInfo(MainJob, subreportHandle,
MySubreportInfo)

'Open the current subreprot in the current section

                subreportJob = PEOpenSubreport(MainJob,
MySubreportInfo.Name)

        Next J

   Next I


End Sub


Private Sub Command2_Click()


'Set the destination of where the report is to be printed (Window, Printer,
Export)

'This would be like the Destination property in the OCX

 Result = PEOutputToWindow(MainJob, "report", 0, 0, 520, 520, 0, 0)

'Start the print job and print it to the window

'This would be like the Action property in the OCX

 Result = PEStartPrintJob(MainJob, True)


End Sub


Private Sub Command3_Click()

'closes the subreport job

PECloseSubreport (subreportJob)

'closes the main report job

PEClosePrintJob (MainJob)
```

```
'closes the print engine before closing the application

PECloseEngine


End


End Sub
```

NOTE	If you know the name of the subreport, you can bypass all of the above code and go directly to PeOpenSubreport(). This will require hard coding the name of the subreport in your application.

Contacting Crystal Decisions for Technical Support

Along with this document, and the *Crystal Reports User's Guide*, we recommend that you visit our Technical Support web site for further resources and sample files. For further assistance, visit us at the web sites below.

Technical Support web site:

http://support.crystaldecisions.com/homepage/

Answers By Email Support:

http://support.crystaldecisions.com/support/answers.asp

Phone Support:

Tel: (604) 669-8379