crystal decisions™

Crystal Reports for Borland C# Builder

Application Deployment

Overview

This document explains how to deploy a .NET windows application that is created in Borland's C# Builder, and integrated with Crystal Reports documents.

Contents

A. Deploying Crystal Reports for Borland's C# Builder

Crystal Reports for Borland C# Builder extends the powerful reporting capability of Crystal Reports to the Microsoft .NET platform. You can use the Crystal Report Designer in Borland C# Builder to create or modify a Crystal report, and then display that report in your Windows or Web application, using a framework-specific viewer control. You can even modify the report dynamically at runtime, by loading the report into the ReportDocument class, and then setting the properties of the ReportDocument instance.

This white paper teaches you how to use InstallShield Express to create a deployment installer project, and to install and then deploy your completed .NET application and reports on a destination computer.

B. Crystal Reports Deployment Components

Certain components are required on the destination computer to achieve a successful installation.

1. Report Files (.RPT)
2. Crystal Reports Merge Modules
3. .NET Framework version 1.1 (a separate installation)

1. Report Files (.RPT)

You can use five different approaches to integrate report files into a .NET web or windows application, three of which are currently supported by C# Builder. Some of these approaches require that the report be imported into the project (embedded), while others reference the report externally by its file path (non-embedded).

When your application is ready to be deployed, you may want to reconsider the decisions that were made at the design stage on whether to use embedded or non-embedded reports. There are advantages and disadvantages to each methodology. If you decide to change your embedding strategy, you will need to refactor your report-viewer binding, before you deploy your application.

The following table compares the advantages and disadvantages to each of the five approaches to binding a report document to the viewer control:

Approach	Description	Procedure	Advantages	Disadvantages
1. Binding to file path of report (not embedded)	The viewer control locates its report source via the file directory path.	Assign the ReportSource property of the viewer control to the file directory path of the external report.	The simplest connection of all.	The report cannot be modified dynamically at runtime; it can only be displayed as originally designed.

Approach	Description	Procedure	Advantages	Disadvantages
2. Binding to ReportDocument class containing file path of report (not embedded)	The report is loaded at runtime, by its file directory path, into the ReportDocument class. This ReportDocument is then bound to the viewer control.	1. Declare ReportDocument as a class variable. 2. Assign a new instance of ReportDocument to the variable. 3. Load a report from the file directory, by passing the file path as a string parameter to the Load method of ReportDocument. 4. Bind the viewer control to the ReportDocument class variable.	1. The report is not embedded. (See next section for details.) 2. The report instance can be modified dynamically at runtime through the properties of the ReportDocument instance, which encapsulate the report.	1. The report is not embedded. (See next section for details.)
3. Binding to ReportDocument class that contains untyped report (embedded)	The report is loaded into the project, generating a report class. The class is upcast to the ReportDocument class. The ReportDocument is then bound to the viewer control.	1. Import the report, which auto-generates a report class that inherits from ReportClass. 2. Declare ReportDocument as a class variable. 3 Upcast a new instance of the imported report to the ReportDocument class variable. 4. Bind the viewer control to the ReportDocument class variable.	1. The report is embedded. (See next section for details.)	1. The report is embedded. (See next section for details.)
Two additional approaches available, but not supported, in C# Builder:				
*4. Binding to a Strongly-Typed Report (embedded)	The report is loaded into the project, generating a report class. The class is then bound to the viewer control.	1. Import the report, which auto-generates a report class that inherits from ReportClass. 2. Declare the new report class as a class variable. 3. Bind the viewer control to the new report class variable.		
*5. Binding to a strongly-typed Cached Report (embedded)	The report is loaded into the project, generating a cached report class. The cached class is then bound to the viewer control.	1. Import the report, which also auto-generates a report class with a prefix on its name of 'Cached' that implements the ICachedReport interface. 2. Declare the new cached report class as a class variable. 3. Bind the viewer control to the new cached report class variable.		

You might decide to change your embedding strategy before deployment. The following section explains how to switch strategies in either direction:

- Switching from embedded to non-embedded report files
- Switching from non-embedded to embedded report files

a. Switching from non-embedded to embedded report files

When reports are added or imported to a Windows or Web application, they are added by default as an embedded resource for the application. That means the report is compiled into the manifest for the assembly, and will not be loaded from a separate report file (.RPT).

Embedded resources allow you to distribute and deploy applications without having to distribute report files separately. The advantage of doing this is that applications do not have to deploy external report files, which may be modified or accidentally deleted by end users. The disadvantage is that if a report must be modified, the entire application must be recompiled and redeployed to save the changes.

What type of application works best with embedded report files? Applications where:

- Reports do not change frequently (or at all)
- Report modification, deletion, or path relocation must be prevented
- Simple deployment is preferred (as few files as possible)

To change non-embedded reports to embedded reports, and refactor the report binding method:

1. Locate each report, by the path given in the code of the application.
2. Import each report into the application.

   Note: For more information on how to import reports, see the Crystal Reports section of the C# Builder help documentation.
3. For each report in the application that was bound to a viewer using one of the non-embedded methods, refactor the code to bind it to the ReportDocument class using the embedded method (untyped report).

   Note: Consult the table on the previous page to understand each approach in detail.
4. Build the project, and test the reports to verify that each report works correctly.

b. Switching from embedded to non-embedded report files

Non-embedded report files are report files that are not compiled into an application's assembly, and therefore are distributed separately from the executable.

When you deploy applications that do embed report files, the report files must be manually added to the destination directory of the installer project. The advantage of keeping reports outside of the assembly is that reports can be easily modified and redeployed, without having to recompile and redeploy the entire assembly. The disadvantage is that reports can be modified, deleted, or misplaced by users in error. A report that is misplaced will not be found by the report file path coded in the application, causing exceptions to be thrown.

What type of application works best with non-embedded report files? Applications where:

- Reports change frequently
- Modification of reports by end users is highly desirable
- Report deletion and path relocation issues are not a concern

To change embedded reports to non-embedded reports, and refactor the report binding method:

1. Locate each embedded report in the application.
2. Copy each embedded report into a common folder in the file directory that is compatible with your deployment plan.

   Note: The reports in your C# Builder project are imported, not embedded—embedding does not occur until compilation, when the reports are embedded inside the windows executable file. Therefore, from the C# Builder project, these report files remain accessible and can be copied to a file directory that you specify.
3. Refactor the code for each report that was previously bound to a viewer using the embedded approach to either of the following non-embedded approaches:

   - Binding via file path of report (non-embedded)
   - Binding to ReportDocument class containing file path of report (non-embedded)

     Note: If you called any properties of the ReportDocument class in your original code (if you have modified any of your reports dynamically at runtime), you must use the ReportDocument approach.

     Tip: If you have multiple reports to be accessed from a common root directory, specify a constant for the root directory path.
4. Build the project and test the reports to verify that each report works correctly.

2. Crystal Reports Merge Modules

Crystal Reports provides merge modules that contain the necessary components to run Crystal Reports in a distributable web or windows application. These merge modules are used with an installer product to ensure the correct report components and component versions are installed with your application.

Downloading modules

To download the merge modules:

1. In a browser, navigate to http://support.crystaldecisions.com/search
2. In the search field, enter cr_net_2003_mergemodules_en.zip.
3. Click **Search**.

   The zip file displays in the browser.
4. Save the zip file to your local hard drive.
5. Extract the files to C:\Program Files\Common Files\Merge Modules\.

Merge module descriptions

Four merge modules must be included in a setup project, to deploy reports.

Merge Module	Description
Crystal_Managed2003.msm	Installs all the CR for C# Builder managed components, such as CrystalDecisions.CrystalReports.Engine.DLL, CrystalDecisions.Web.DLL, and CrystalDecisions.Windows.Forms.DLL.
Crystal_Database_Access2003.msm	Installs all database drivers that the report uses to connect to various types of data sources. This merge module also installs export destination and format drivers, which can save reports in different file formats, such as RTF and HTML. This merge module installs all non-managed runtime components for CR for C# Builder, including the charting components.
Crystal_Database_Access2003_enu.msm	Installs language-specific (localized) components. Some of the localized charting and exporting (like PDF) components are installed by this component. Each language the product is released in will have a specific version of this merge module.

Crystal_regwiz2003.msm	Configures registration information to track licensing on deployed computers (server or client). Crystal_regwiz2003.msm exposes a LicenseKey property when it is added to a setup project. You must set that license key before you build the setup project. The 19-digit license key is emailed when the product is registered. Enter it into the LicenseKey property for the Crystal_regwiz2003.msm merge module. **Note:** If the license key is not set for the Crystal_regwiz2003.msm merge module, an error occurs when building the project. If the setup is installed on a target computer without specifying the license key for the merge module, various "keycodev2.dll" errors occur.

3. NET Framework version 1.1 (a separate installation)

Since Crystal Reports for C# Builder is based on version 1.1 of the .NET Framework, **at minimum version 1.1 of the .NET framework is a pre-requisite on the destination computer**.

It is recommended that the .NET framework be installed separately from the deployment installer file on destination computers. Version 1.1 of the .NET framework can be installed either from Windows Update or downloaded as the executable dotnetfx.exe from this url:

http://msdn.microsoft.com/netframework/downloads/redist.aspx

C. Deploying a Windows Application

This section shows you how to deploy a .NET Windows application, created in C# Builder, that integrates Crystal Reports. It explains how to use InstallShield Express to deploy an application. You can download a trial copy of InstallShield Express from their website at http://www.installshield.com.

Preconditions

- You have created the .NET Windows application in C# Builder.
- You have added or imported to the application reports that were built in Crystal Reports, and bound them to the viewer control.

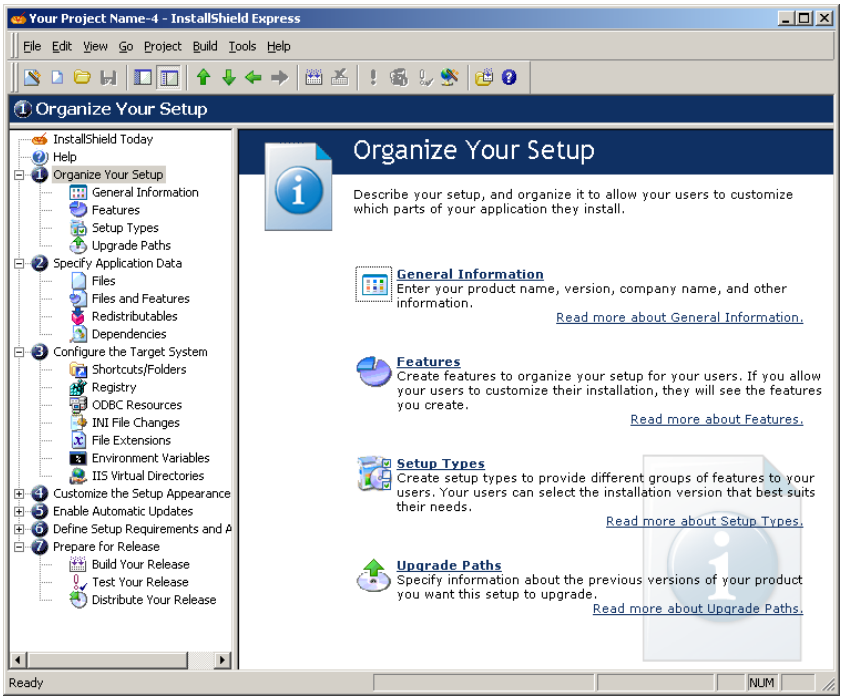
  Note: The reports may be either embedded or non-embedded. For more information on embedded and non-embedded reports, see section B.
- You have compiled and tested your application, and verified that both the application and its reports work correctly.
- You have installed and tested InstallShield Express.

To setup your deployment project using InstallShield Express:

1. From the **Programs** menu, select **InstallShield / Express**.

   The InstallShield Express dialog box appears.
2. Select **Create a new project** on the left pane.
3. Select **Blank Setup Project** in the right pane, and click **Create**.

   The Organize Your Setup window appears. The left frame contains a tree view of seven major topics on how to configure your installer file.



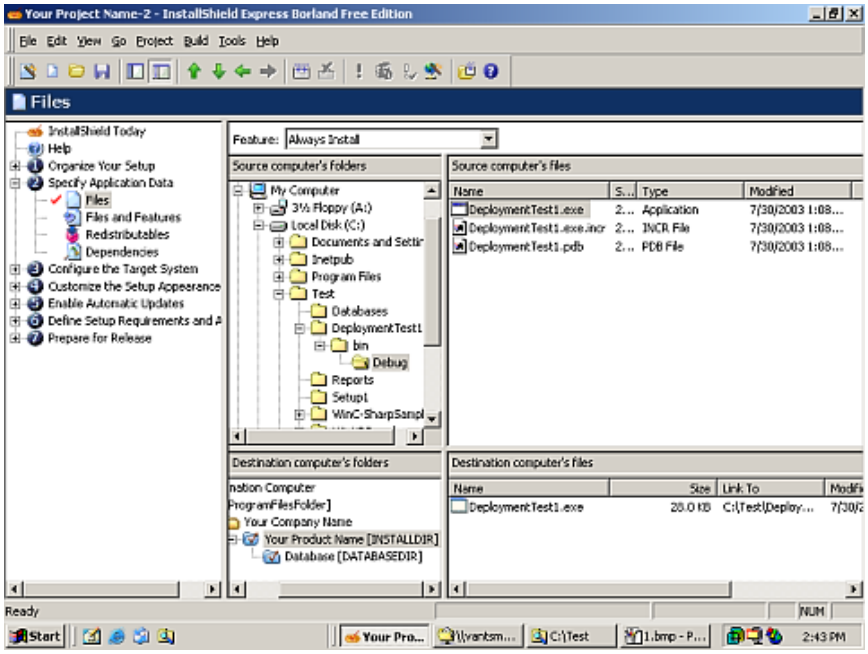
4. In the tree view, expand **Organize Your Setup to** select **General Information.**

   The General Information view appears in the right frame.

5. Enter all general information for the installer (such as the product name).

6. In the tree view, expand **Specify Application Data** to select **Files**.

   The Files view appears. The upper explorer pane shows the directory of your local computer. The lower pane shows the directory of the destination computer.



7. Locate the project directory for your C# Builder windows application project.

8. Within the directory, navigate to the /bin/Debug/ directory, to find your Windows executable file.

   Tip: If you cannot find the Windows executable file at the /bin/ or /bin/Debug/ levels, recompile your project in C# Builder.

9. Drag the application executable (.exe) file to the product directory of the destination computer, in the lower pane of the **Files** view.

This completes the file selection for an application whose reports are embedded. To complete the file selection for an application whose reports are non-embedded, complete the following section.

To add non-embedded reports to the destination computer:

1. In the upper pane of the **Files** view, locate the directory(ies) containing your .RPT files.

2. In the lower pane, recreate the report directory path(s) in the directory of the destination computer.

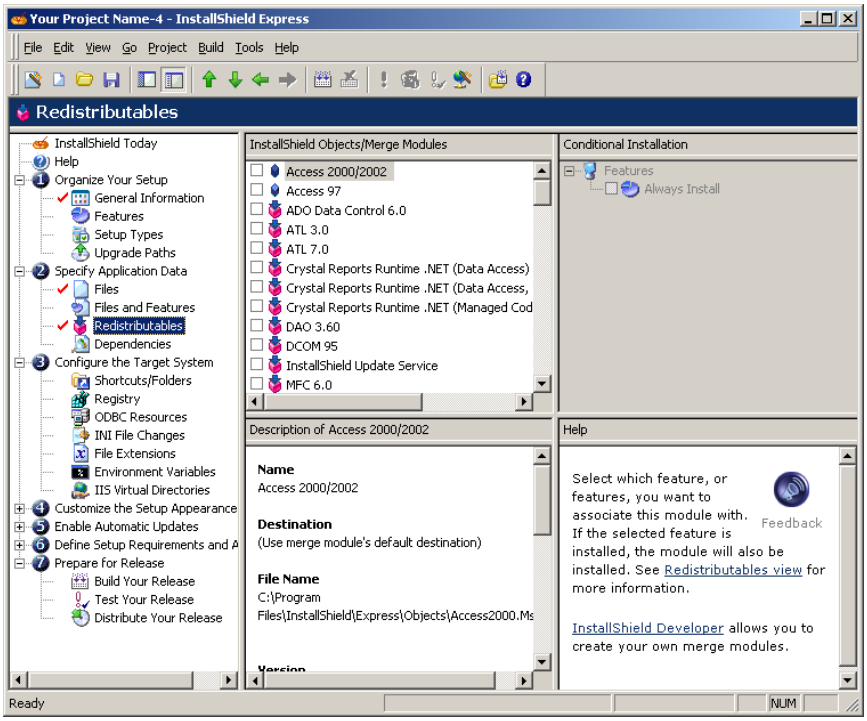
   Note: The directory path specified in your code for non-embedded reports must match the directory you create on the destination computer, to have reports load successfully.

3. Drag each report into its corresponding directory on the destination computer.

To add the merge modules:

1. In the tree view, expand **Specify Application Data** to select **Redistributables**.

   The Redistributables view appears. Note that the merge modules you placed earlier into C:/Program Files/Common Files/Merge Modules/ are now visible as check boxes in the upper left pane of the Redistributables view.



4. Check the three Crystal merger modules check boxes.

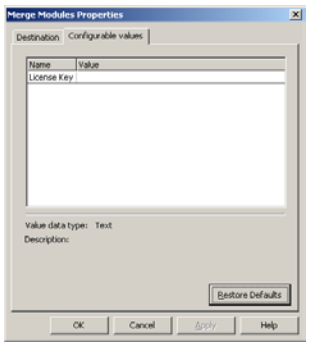
   Note: The display names are different from the file names of the merge modules. Those names are:

   - Crystal Reports Runtime .NET (Data Access)
   - Crystal Reports Runtime .NET (Data Access, English)
   - Crystal Reports Runtime .NET (Managed Code)

   To prevent accidental selection of earlier versions of these Redistributables (which will cause an installation failure) you must verify that the .msm file name for each file matches the files you downloaded earlier. Verify the file names in the Description pane of the Redistributables window.

5. Scroll down and check the **Seagate Registration Wizard** check box.

   After you check **Seagate Registration Wizard,** the **Merge Modules Properties** dialog box appears.



6. Locate your license key for the Crystal Reports plugin.

   Note: This was the non-expiring product code (**not** the registration number) which you received by email upon registration of Crystal Reports for C# Builder.

7. Enter the license key (non-expiring product code) into the Value column in the **Merge Modules Properties** dialogue box, and click **OK**.

8. On the **File** menu, select **Save**.

   Your installer application is saved. You are now ready to build your application.

9. On the **Build** menu, select **Build Single Image**.

   The build process begins. This may take some time. When the build process is complete, see the next section for instructions on how to test your build.

To test your build:

1. On the **Build** menu, select **Test Single Image**.

   This runs a test of your install, and locates . any problems.

   Note: If you encounter problems with your install file that are not related to Crystal Reports, please contact the installer application vendor.

2. Verify that a **Setup.exe** file has been created your deployment project.

   Note: The default location for the Setup.exe file using Install Shield Express is:

   /My Documents/MySetups/YourProjectName-#/Express/SingleImage/DiskImages/DISK1/setup.exe

3. Verify that version 1.1 of the .NET Framework has been installed on the destination computer.

4. Copy the Setup.exe file to a destination computer.

5. Run Setup.exe to install your test application.

6. Locate the test application on the destination computer located at file path:

   /Program Files/Your Company Name/Your Product Name/YourApplication.exe

7. Launch the application and verify that the application is working and that the Crystal Reports are displaying correctly.

D. Deploying a Web Application

This white paper does not include instructions on how to create an installer for a web application created in C# Builder, because it is assumed that in most cases the application would be deployed on a web server at your location.

E. Conclusion

This concludes the instructions for how to deploy a .NET application, created in C# Builder, that includes reports created in Crystal Reports. For additional information and updates, please visit the following site:

http://www.crystaldecisions.com/getinthezone

F. Contacting Crystal Decisions for Technical Support

We recommend that you refer to the product documentation, or visit our Technical Support web site for more resources.

Self-serve Support:

http://support.crystaldecisions.com/

Email Support:

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

Telephone Support:

http://www.crystaldecisions.com/contact/support.asp