By Bill Todd

ReportBuilder Server Edition 7

Multi-tier Reporting

Do you need an easy way to implement reporting in your multi-tier applications? Do you need to publish reports on the Web? If your answer is "Yes," then ReportBuilder Server Edition 7.0 is the tool you've been looking for.

Moreover, if your existing reports were created using "regular" ReportBuilder, converting them to a report server is a snap. And even if you've been using another reporting tool, the ease and power of ReportBuilder Server Edition will make you want to change.

Multi-tier Reporting

First, let's look at reporting in multi-tier applications. If you've ever tried to run a report from a DataSnap client, you know what a problem it can be. If the amount of data required to print the report is small, the solution is easy. Just load the data into a ClientDataSet and use that as the data source for the report. If the volume of data is large, bringing the data to the client may cause an unacceptable traffic load on the network, and/or unacceptable memory consumption on the client.

Network traffic can be a particular problem if the DataSnap client and server are communicating across a low-speed WAN. (There's another solution, but it requires additional custom code. You can print the report to a file on the server. By adding custom methods to the *IAppServer* interface you can compress the file, send it to the client, uncompress it and save the file to disk. Then the client can print the file.)

Using ReportBuilder Server Edition you can easily write a report server that lets client applications request a report. When the server gets a request, it generates the pages requested by the client, compresses the pages, and returns them to the client across any TCP/IP network. The client application decompresses the report, then lets the user preview or print it just as if the report were being run locally.

To minimize network traffic, the client automatically caches the pages so they don't have to be sent across the network if they're requested again. The server also maintains session

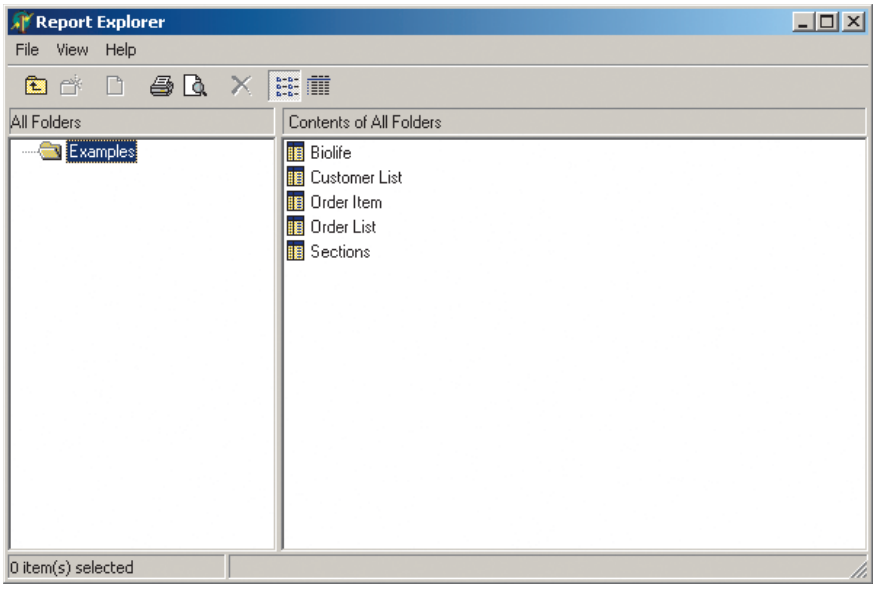

Figure 1: ReportBuilder's report explorer.

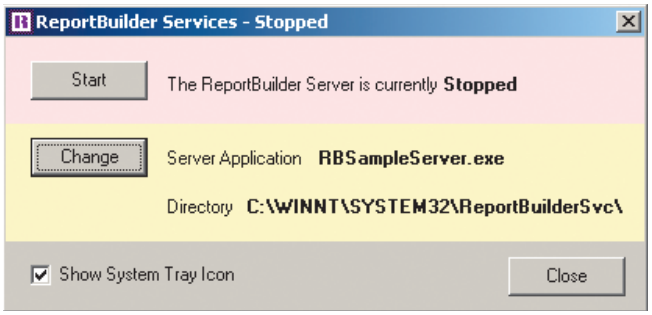

Figure 2: The ReportBuilder Services dialog box.

information for each client, so it knows which pages have already been sent. The combination of data compression, client-side caching, and persistent session information on the server reduces network traffic to the minimum possible.

Converting to the Server Edition

If you've been using ReportBuilder, your reports fall into one of three categories. They're compiled into an application, stored as report templates in a file or database, or stored in a report archive.

Compiled into an application. If you have an application that includes one or more reports that are compiled into the program, moving them to a report server requires just a few simple steps. Start by creating a new application. Then move the form that contains the report to the folder that holds the new application and add the form's unit to the new application. If the data access components that provide the data for the report are in a separate data module instead of on the form, copy or move the data module to the new application's folder, and add its unit to the application.

Drop an rsServer component from the RBServer page of the Component palette onto the new application's main form. By default, the server and client communicate on port 1333. Change the *Port* property if you need to use a different port. Add an *OnCreate* event handler to the form and add the following statement:

Just the Facts

ReportBuilder Server Edition 7.0 makes it easy to implement a complete multi-tier or Web-based reporting solution. The combination of data compression, client-side caching, and persistent session information on the server reduces network traffic to the minimum possible.

Digital Metaphors Corp.
11001 W. 120th Avenue, Suite 210
Broomfield, Colorado 80021

Phone: (303) 531-8032
Fax: (303) 531-8036
Web Site: http://www.digital-metaphors.com
Price: US$999 for ReportBuilder Enterprise Edition, one development license, and one CPU deployment license. US$249 per additional CPU.

```
rsServer1.Active := True;
```

Setting *Active* to *True* causes the rsServer component to begin listening for client connections.

Each report must be registered with the server. To register a report, open its unit and add rsReportCatalog to the **uses** clause in the **implementation** section. Next, add the following call to the unit's **initialization** section.

```
TrsReportCatalog.RegisterReport('Examples',
  'Customer List', 'ppReport1', TfrmOO1CustomerList);
```

The first parameter is the volume the report will be in. Think of volumes as folders; they're a way to group and organize reports. The second parameter is the name you want displayed for this report when you view the registered reports using the rsClientReportExplorer. The third parameter is the name of the TppReport component, and the fourth is the class name of the form that contains the TppReport.

The last step is critical. You must make sure that the data access components used by the report are thread safe. How you do this depends on which data access components you're using. For example, if you are using the BDE (Borland Database Engine) you must use a Database and a Session component, with the Session component's *AutoSessionName* property set to *True*. With the InterBase Express components, you must use a remote connection.

Building a client application is even easier. Create a new application, then add rsClientReport and rsClientReportExplorer components to the main form. Set the *ClientReport* property of the rsClientReportExplorer to *rsClientReport1*. Add a button to the form and create an *OnClick* event handler for the button that calls the rsClientReportExplorer's *Execute* method. Run the server, run the client, click the button and you will see the Report Explorer dialog box shown in Figure 1. You can select any report from the report explorer, and preview or print it.

Stored in template files. If your ReportBuilder reports are stored in template files, building a server application is just as easy. The only difference is that you drop an rsReportTemplateVolume component on the server's main form or data module, and set its *FileDirectory* property to the folder that contains the templates. If the templates are stored in a database table, set the *StorageType* property to *stDatabase*. Next, expand the *DatabaseSettings* property and set *DataPipeline* to the *DataPipeline* component that supplies data from the template table, set *BLOBField* to the field that contains the template, and set *NameField* to the field that contains the name of the report.

Stored in a report archive. If you have reports in a ReportBuilder archive, creating a server requires just two steps. Add an rsReportArchiveVolume component to the main form, and set its *FileDirectory* property to the folder that contains the archive. Second, drop an rsServer component on the main form, and create an *OnCreate* event handler that sets its *Active* property to *True*.

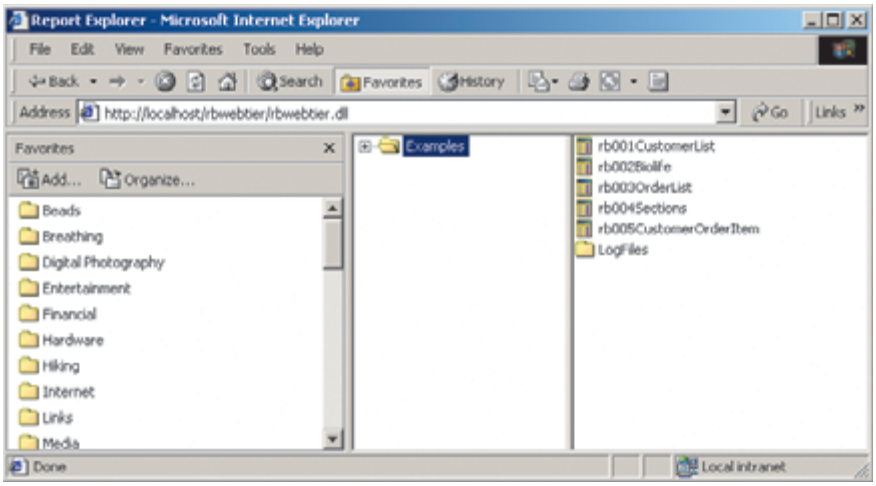

Figure 3: The report explorer in a Web browser.

One problem with the browser-based report explorer is that it looks too good. My first reaction was that it must be an ActiveX control. Surprisingly, that isn't true; the report and the explorer are created with XHTML and JavaScript generated by the server.

When serving reports on the Web, you may have a large number of users trying to run reports at the same time. ReportBuilder Server Edition handles this through its support for report server farms. The *ServerFarmSettings* property of the rsWebTier component lets you enter the IP addresses of the servers in the farm, and choose either minimum-load or round-robin load balancing.

An obvious problem with the servers described so far is that they are Windows executables. That means that someone has to log on to the machine where the server will run and execute the server program. Having a user logged on to a file, database, or Web server machine creates a serious security hole. The solution is to write the report server as a Windows service. Fortunately, you don't have to: ReportBuilder Server Edition includes a Windows service, RBWinService.exe, that can automatically run a report server application.

Using RBWinService.exe, like everything else in ReportBuilder Server Edition, is remarkably easy. First, run the EXE to install the service. Next, add the rsServerActiveX unit to the **uses** clause in the **interface** section of the server application's main form. Next, compile and run the report server with the `/regserver` command-line switch. Double-click the ReportBuilder service icon in the system tray to display the dialog box shown in Figure 2. Click the **Stop** button to stop the service; then click the **Change** button and select the server application that you want to run. Finally, click the **Start** button and close the dialog box. Now your server will run automatically each time the host machine is started.

Reporting on the Web

Once you have a report server running, publishing your reports on the Web is a matter of writing an ISAPI.DLL that contains one component. Start by creating a new ISAPI application using the Web Server Application wizard in the Object Repository. Save the application. Create a virtual directory in IIS that points to the directory that will contain the ISAPI DLL. Create a folder under the folder that contains the DLL to hold cached report pages.

Now drop an rsWebTier component on the application's Web module. Next, set the *CacheDirectory* property to the folder that will hold cached report pages. Set the *WebCachePath* property to the URL of the cache directory. Set the *WebModuleURI* property to the URL of the folder that holds the DLL. Compile the DLL, then call it from your browser. You'll see the report explorer shown in Figure 3. ReportBuilder also supports ASP and Apache Web modules.

To achieve accurate printing from a Web browser you can add either of two third-party export components to your Web tier project. After you add the export component a **Print** button will appear in the report preview window in your browser. When you print, the report is exported to Adobe Acrobat PDF format.

Documentation

ReportBuilder has always had superb documentation; ReportBuilder Server Edition continues this tradition. In addition to online help and sample applications, you get a 66-page tutorial that takes you through building each type of server application step by step. The tutorial starts by converting an application with form based reports to a server and takes you through template- and archive-based reports, using the ReportBuilder Server Edition service and setting up a report server farm.

Pricing and Conclusion

ReportBuilder Server Edition is priced at US$999. This includes ReportBuilder Enterprise Edition, one development license, and one deployment license. The deployment license lets you deploy to a single CPU server. Additional deployment licenses are available at US$249 per CPU.

The most impressive thing about ReportBuilder Server Edition 7.0 is how easy it is to use. All you need to do to implement a complete multi-tier or Web-based reporting solution is add a couple of components to a project, set a few properties, and add one or two lines of code. The excellent tutorial makes getting started a snap. Even configuring report server farms is easy. This well-designed, well-documented, easy-to-use product will meet all of your reporting needs. It's easy to see why ReportBuilder has won Best Reporting Tool and Product of the Year in the last four *Delphi Informant* Readers Choice Awards.

Bill Todd *is president of The Database Group, Inc., a database consulting and development firm based near Phoenix. He is co-author of four database programming books, author of more than 100 articles, a contributing editor to* Delphi Informant Magazine*, and a member of Team B, which provides technical support on the Borland Internet newsgroups. Bill is an internationally known trainer and is a frequent speaker at Borland Developer Conferences in the United States and Europe. Readers may reach him at bill@dbginc.com.*

New

ReportBuilder™ Server Edition

Are your reports like a message in a bottle?

After slaving over hundreds of reports and spending countless hours getting them just right, you realize that all of this wonderful 'content' is trapped in a teeny tiny LAN. You know it's the modern day equivalent of the message in a bottle, but after getting ample practice in the art of 'suffering fools' at the latest meeting with your CEO, who remains incredulous that these reports 'aren't in my browser', you realize that you better get the message out of the bottle and quick. One problem: creating a report server isn't exactly a cakewalk.

Enter **ReportBuilder™ Server Edition**. Take those reports, in whatever configuration they may currently reside, register them with this report server, and it's a done deal. Use the WebTier component to quickly deploy everything via your Web Server, or use the ReportServer component to deliver reports across the LAN or across the Web to a thin client. Either way you realize that you've just found something that's going to save your bacon. You want to know who to thank but the guy at Digital Metaphors just says '*Don't worry about it, it's what we do.*'

Report Server

- Serve report content over the web via a multi-threaded NT service
- Provide a list of available reports in a folder tree structure
- Send and receive search criteria for a given report

Web Tier

- Publish reports as XHTML to web browser clients
- Build your own ISAPI, ASP, Apache or CGI web applications
- Customize the generated XHTML and JavaScript framework

Client Support

- Web: Select available reports, enter search criteria and preview report content via IE 4, IE 5, Netscape 4, and Netscape 6
- Thin Client: Use the standard ReportBuilder UI to access server-based reports from within a Delphi application.

ReportBuilder Server Edition. *Get the message out of the bottle.*

• DIGITAL • METAPHORS™

2002 READERS CHOICE Delphi INFORMANT MAGAZINE Product of the Year

2002 READERS CHOICE Delphi INFORMANT MAGAZINE Best Reporting Tool

For a live demo surf: www.digital-metaphors.com/server