Java Programming for Internet and Web Development using IDS

Details the process of building an Internet discussion group application around Smithware’s new I*Net Data Server and Btrieve using Java.
— Ken White

Going Public: Connecting your Database with the World Wide Web

Outlines a number of options for making database information available on your web site.
— Howard Shirley

Making ODBC Work

The ODBC interface from Pervasive Software can give you a lot of options, you can just get it installed and working properly. Here are a few of the common pitfalls and how to get around them.
— Richard Trocino

When to Use Scalable SQL, When to Use Btrieve?

Pervasive Software currently offers two very flexible and useful databases, Btrieve and Scalable SQL. This article helps developers decide which one is most appropriate for their particular application.
— Richard Trocino

New Btrieve Newsgroup on Usenet

Btrieve Developer’s Journal announces the comp.databases.btrieve newsgroup on Usenet, and discusses how to get on-line.
— Kathy Jones
Smithware ActiveX Controls for Btrieve makes it easy to write powerful client/server database applications in any programming language or development environment that supports industry-standard ActiveX (OCX) technology.

Code Avoidance Technology (CAT) Says It All. Developed exclusively by Smithware, CAT means we have written lots of code so you don’t have to. We've made using Btrieve as simple as dragging and dropping a control onto a form. CAT makes adding ActiveX Controls for Btrieve to your project easy.

Standard properties, events and methods connect your application to your data. Simple point-and-click setup of client/server-extended operations and file joins let you implement high-performance features effortlessly.

You can build a simple data browser without code, just by dropping a text box, list box, and scroll bar on a form! Our comprehensive tutorials and sample code get you started fast.

Data Aware Controls Complete Your Application's Data Interface.

Our Controls Feed Your Application’s Need for Data. The VAccess data source control is the direct property, event, and method interface to the Btrieve engine.

Each VAccess control on a form accesses a data file. The VAccess control has properties that determine which file is opened and how data is retrieved and updated. It also has methods that allow you to perform operations on the file as well as events that allow you to process your own code scripts at predefined points in project execution.

Multiple VAccess controls can be joined together to maintain relationships between tables automatically. The VACommandButton control allows users to insert, update, delete, and retrieve records. VAScrollBar controls can be bound to VAccess data source controls to enable record navigation through Btrieve tables. Consistent use of indexes and integrated Btrieve percentage operations track the application’s current position within a table.

The VACheckBox and VAOptionButton connect the application to logical fields. The VAComboBox can be bound to one field in a table, allowing the user to select a value for the field from a predetermined set of values. It can also serve as a record list control, or even as a list of available index fields for user navigation.

The VAListBox can serve as a multi-column record list control. The VATextBox can be used to update individual fields in a table, and also can be used in search mode, automatically seeking matching records from a table as text is entered by the user.
These Things Really Move!

Smithware ActiveX Controls for Btrieve gives you proven performance advantages over other database access methods.

Benchmarks performed using a series of tests published in **Visual Basic Programmer's Journal** (May 1996, Vol. 6, No. 6) demonstrate the outstanding performance of our ActiveX Controls for Btrieve. In a suite of tests involving indexed record retrieval, record inserts, updates, and deletes, sequential file scans, and joined files, Smithware ActiveX Controls for Btrieve consistently outperformed all of the other Btrieve database access methods.

We give you unprecedented control over the data management tasks your application performs with the option to enable functionality you want to use, or to disable functionality you don’t need.

You never have to face an hourglass cursor while the program goes off on a wild goose chase through your files - a common complaint of developers using ODBC, the VB Jet database engine or other data access methods.

We’ve further optimized performance by allowing you to enable or disable event calls, letting you eliminate unnecessary calls to empty events in your code. Table information stored with the application means faster load times and less complexity at run time. Just another way to boost your application’s performance.

We Speak Your Language.

In addition to providing unbeatable performance, our advanced data binding gives you an advantage not available from any other database development tool.

Because Smithware ActiveX Controls for Btrieve uses industry-standard ActiveX (OCX) technology, developers can choose from a wide range of languages and development environments for their applications. ActiveX Controls for Btrieve are supported in such popular development environments as Microsoft Visual Basic, Borland Delphi, Borland C++ Builder, Microsoft Visual C/C++, Microsoft Internet Explorer, and more.

Take Your Application to the Internet.

All applications developed with Smithware ActiveX Controls for Btrieve are Internet-enabled applications. No other database access tool lets you take your application to the Internet without changing a single line of code!

With Smithware ActiveX Controls for Btrieve, the application that runs on a desktop at the office is the same application that can run on a laptop on a salesman’s tray table at cruising altitude, on the convention floor in Las Vegas, or in a hotel room in London, providing users with transparent client/server access to their important corporate data using the power of the global Internet. For more information on our Internet Database Solution visit [http://www.smithware.com/ids](http://www.smithware.com/ids).

Talk To The Cat That’s Got Your Tongue.

Ask About ActiveX Controls For Btrieve with Code Avoidance Technology.

It’s all here! Proven, high-performance Btrieve database access. The ease of use you’ve been looking for. Supported by all of the most popular programming languages.

Make Smithware’s ActiveX Controls for Btrieve part of your development language today by calling 800-828-7438. We speak your language.

Smithware, Inc.
2416 Hillsboro Road, Suite 201
Nashville, Tennessee 37212
615-386-3100
Fax: 615-386-3135

http://www.smithware.com
e-mail: info@smithware.com

What you get: ActiveX Controls for Btrieve data source and bound data controls for 16-bit and 32-bit application development. Smithware’s DDF Builder, the number one data definition tool used by thousands of developers worldwide. Sample code, tutorials, and on-line developer’s reference. Btrieve version 6.15 Client Engines for Windows and Windows NT. Compatible with Microsoft Visual Basic, V4, V5, Borland Delphi, Borland C++ Builder, Visual C++, Internet Explorer 3.0, Optima ++, Visual Fox Pro, and other programming languages and development environments which support ActiveX (OCX) custom controls. All Smithware CATs are spayed, neutered, housebroken, and guaranteed pedigreed.
OWN THE GAP.

With completely new Scalable SQL 4 from Pervasive Software, you can own the gap between entry level and enterprise.

Somewhere between entry level and enterprise, there’s only one embeddable, relational database that has all the answers: new Scalable SQL 4 from Pervasive Software. A small memory footprint. Ease of use. Scalability and high performance. In short, all a small to mid-size business needs. Except the one thing it doesn’t need: a database administrator.

Features include everything necessary to build applications for the middle market. Inscribe—Pervasive’s new Visual Basic™ compatible scripting tool, stored procedures, triggers, cost-based optimization, and transactional logging. Check us out at http://www9.scalablesql.com. And while you’re there, be sure and play our “Name That Gap” Web game and register for a $2,000 cash giveaway.

OWN THE GAP

Pervasive Software Inc., Btrieve, Btrieve 6, Scalable SQL, ODBC Interface, Inscribe, Client Server in a Box and their designs, and Built to be Bundled are trademarks or registered trademarks of Pervasive Software Inc.

©Copyright 1997 Pervasive Software Inc. All rights reserved worldwide.

800-287-4383
http://www9.scalablesql.com
US 512-794-1719  France +33-1-46-69-7955  Germany +49-6172-925830

Limited Introductory Offer: Scalable SQL 4 Developer Suite just $595 (a $1,295 value!)
Java Programming:
I* Net and Web Development
with IDS

By Ken White
ken@smithware.com

As most faithful BDJ readers are aware, Smithware is approaching the launch of its newest product, the I*Net Data Server. As one of the primary programmers responsible for this product, I’m genuinely excited about the prospects of full Btrieve connectivity across the I*Net; I’m even more excited about the ability to use virtually any development environment to write an I*Net application. The ability to convert legacy applications to use the Internet as a LAN or to write new applications that function identically over the LAN and the Internet is, in my humble opinion, something that we’ve needed for a long time.

For this article, however, I did not get to choose the development language. No, we needed a strong sample application to demonstrate the Java Class Library for the IDS, and (more importantly, to you) Kathy, the reigning BDJ editor, needed an article about new development technologies applied to Btrieve. “Very well,” thought I, “I’ll write a discussion group applet which accesses Btrieve in Java using our new class libraries, and then we’ll post the applet and use it to further discuss intranets and the Internet (the I*Net) as they relate to Btrieve. We could even write about it for BDJ!” (Coincidentally, I had this idea immediately after my boss had the idea for me.) Did I mention the “idea” came two weeks before the deadline for the article?

Using traditional HTML methods, you may or may not have a graphical view of the message headers. Also, when you go from a message header to a message and back again (using the “Back” button on your browser), you have to re-load the page that you just downloaded. If you want a longer message history or another method of sorting the headers, you have to get the complete message header list again, if the option is offered at all. Posting a message requires yet another page and a couple of additional “Back” button clicks when done. In addition, all of the work is done on the server, so other users can cause a significant lag in the server’s response to your query. You’ve essentially turned your new $3,000 MMX P200 into a 1979-vintage dumb terminal that can display pretty pictures and parse cookies. Wow! That’s efficient. Plus, you get prescribed tools and pre-ordained processes, all of which may or may not work together, and none of which you may be familiar with. Did I mention just how much memory that browser eats up?

This applet needed to be different. It is a true client/server implementation, and it uses multithreading to download records from the server while other tasks are being performed. It can sort and cache messages on the client side, and it is smart enough to get only the message headers it needs based on the requested history window and the current cache contents. It also registers users with the group and provides for guest (read-only) access.

After working out the general specifications of the applet, I decided to nail down the database design. There were to be three tables: One for users, one for message headers, and one for message detail. By using extended operations, the message headers could be downloaded pretty easily during registration and login. The detail would be downloaded as needed, with a maximum (pre-compressed) message size of 4K. All three of the files would be owner-name protected, and the header file would have an approval flag; only approved postings would be available for download.

The user interface was determined to some extent by the tools available. Since we had recently purchased a copy of Symantec’s Visual Café, I decided to use it for all GUI work. It comes with several auxiliary classes, one of which is the tabbed dialog class. “Perfect!” thought I and laid it out on the form. To protect the resolution-impaired, I decided to make sure that the applet could be viewed inside a toolbar-heavy browser running on a screen in 800x600 mode. Since I like the Windows TreeView control, I decided to use the Symantec TreeView class to display the message headers. The freshly downloaded applet is shown in Figure 1.

Lest you, wary reader, begin to think this a laudatory exposition on the virtues of Visual Café, I must make mention of another package that was actually the primary development tool — Visual J++ 1.1. VJ is much easier to work with and debug, but it has no UI tools of note. Since Visual Café changes the Java code to reflect changes made in the graphical designer, I was able to open and change the source files across development environments with impunity. Visual Café, bless its heart, just has too many problems for full-time use right now, as far as I’m concerned. The graphical layout manager gets confused too easily, the controls are often poorly documented (although, one could argue, not as poorly documented as the language itself), and it is s-l-o-w. For my money, Visual J++ is the best single Java tool I’ve found. Since

See IDS, page 7

Day 1: Building a Better Mousetrap

There were several specifications that had already been decided upon. First, since our site is configured as an open site (meaning the IDS does not require a password for Btrieve access), I had to have some way of limiting access to the files. Second, the discussion group needed to be moderated to prevent obscene, malicious, deceptive, or just plain unfriendly posts. (I certainly didn’t want someone masquerading as Colonel Micro to post a message berating Microsoft when I know for a fact that the Colonel loves nothing more than a new MS product — especially v1.0 of a new MS product.)

Finally, this applet had to demonstrate why the IDS is a better solution than other options. Since this sample used the Java classes, it was competing against other web options. I’ll try not to get too preachy here, but the alleged “other web options” make about as much sense as racing a Pinto in the Indy 500: You’re going to crash, and it’s going to hurt. Let’s look at how a typical web-based discussion group might work.

The discussion group applet described here can be accessed online at http://www.smithware.com/discussion/idsdiscussion.html.

See this Java applet in action — www.smithware.com/IDSDiscussion/IDSDiscussion.html
to apply the most "current" technology (SQL, ODBC, Java, ActiveX, the Internet...) to what is sometimes considered to be "legacy" technology. This issue goes a long way, in my opinion, to illustrate that Btrieve is still solid technology and fills a real need in the development marketplace.

Smithware’s Ken White does an entertaining job of introducing the new Smithware I*Net Data Server and the I*Net Developer’s Kit by going step-by-step through the construction of a Btrieve-based Java applet. Richard Trocino took on my challenge to demonstrate to an idiot (that would be me) how to successfully setup and use the Btrieve ODBC Interface — and he actually survived the process! (Another BDJ contributor, who shall remain nameless, almost resigned in disgust as a result of that challenge! And I really can’t blame him...) In another article, Richard considers when it is appropriate to use Btrieve, and when to use Scalable SQL.

Doug Reilly shows how Btrieve and Scalable SQL can coexist in one environment and introduces the new version 6.3 of the Btrieve interface, and Craig Bobchin reviews the latest release of Smithware Crystal Reports for Btrieve.

I also think you will like what we have planned for our next issue as well. Jim Kyle will be looking into a new buzz-word in the Pervasive Software vocabulary, “embeddable database” seeing where Btrieve fits in. Jim also will give us a history of ODBC and how it applies to Btrieve and Scalable SQL. Richard Trocino will take a closer look at the actual performance of the Pervasive ODBC Interface, and new directions like JDBC and OLE-DB. We’ll also research “The Gap,” where Pervasive sees their primary market these days.

So stick around, there’s plenty more Btrieve technology news and informatio...
this wasn’t my money, I chose to use both tools — Visual Café for UI and some event handling and Visual J++ for “real work” and debugging.

After getting the UI for the first panel done, I came across a design issue I had not addressed sufficiently. Normally, I prefer normalization of data. However, normalization of data in the context of low-bandwidth can hurt more than it helps. The design issue in question was the link between message headers and users — I’d like for user names to stay current (i.e., normalize the data), but getting another redundant record for each message header is inefficient when bandwidth and packet routing is at a premium, as is the case here. So, I decided to do an extended operations get on all users at applet startup, and search that cache on the client side for user names. This way I could remain normalized and put most of the cost on the client’s processor, which in a web browser is the cheapest resource available. Of course, the class used to get all users must be thread-safe, but this is relatively easily accomplished in Java.

After about three hours, I was ready to try to run the applet — and what do you know, it worked! Well, at least the first dialog page did. With that done, I found myself distracted by worldly pleasures such as Smithware ActiveX Controls enhancements and Btrieve configuration problems. 

Day 1 Time: 3.5 hours.

Day 2: Threading Threads

Day two was largely spent on the implementation on the helper threads of the applet, of which there are three. The first, mentioned previously, downloads a cache-full of users for normalization purposes. The second is more elaborate — it downloads and/or sorts the message headers on an as-needed basis. The third downloads a single message body and was implemented later in the project.

For the first thread, I accepted a limitation on the cache size — only the first 800+ users will appear in the cache. The second thread, however, needed a cache that was limited only by system memory. Using multiple extended get calls with array copies/resizing achieved the desired result while minimizing applet cost over low-bandwidth connections. In addition, by keeping track of the current cache, I made sure that the thread never gets the same record twice. The strength of Java’s OOP approach is demonstrated by the fact that the calling thread does not need to know the contents of the header thread’s cache.

The rest of this day was spent debugging and testing. One of the problems with Java is the amount of time it takes to fire up a browser or applet viewer and get the applet running. Even on a P200, I found myself waiting between one and two minutes for a run to start or stop.

Day 2 Time: 3 hours

Day 3: Kick Me!

Now we come to the only part of a developer’s journal I don’t like — I can either “creatively omit” my stupid mistakes or describe them in vivid detail, making myself the object of ridicule for strangers and comrades. Suffice it to say that I am not likely to forget any time soon the number of milliseconds in a day.

After solving the nefarious millisecond problem, I ran into another tangle. The applet worked fine in debug mode if I stepped through the code, but crashed hideously when I tried to run it in either debug or release mode. Although this sounds more painful than the last Police Academy movie, it was actually not that hard to fix. I determined that it had to be a timing issue, which led me directly to a threading problem. Conflict resolved — I began to see header data.

The next issue I had was that of re-enter calls to the message header thread. Although a thread that has exited the run() routine still exists in memory, it cannot be started again. Since there was a fair amount of persistent information in the message header thread, re-initialization was not an option. Instead, I put the thread in a semi-infinite loop. Before you start to critique my terminology, let me say that I realize that technically a loop is either infinite or it is not. However, this loop is in itself infinite, but when multi-threading comes into play, all bets are off.

As the code demonstrates, if left to its own devices, this thread would run forever were the suspend() statement removed. However, we’re pulling the strings from the calling thread and we’re not going to let that happen. First of all, the thread executes once and then stops and waits because of the suspend() call. If the calling thread wants to restart the thread, say, for a refresh, it simply calls thread::resume() to run the thread one more time. When the calling thread wants to kill this thread, it simply calls thread::KillMeNow() before calling resume(), after which the thread dies of natural causes. In this way the thread can be called any number of times but retain all of its state information.

Day 3: 4.5 hours

Day 4: Show Me the Cache

This was the day to implement the biggest optimization of the project — the cache handling of the message header thread. To do this, the thread needs to keep up with the current requested history and the total number of days in the cache. If this information is known, getting any missing data is as easy as setting up an extended operations call and performing it, about ten lines of code. By using Java and communicating directly with the Btrieve data via the IDS, true client/server computing becomes possible, and optimizations can be made that would not have been possible in a “normal” web page.

This went as well as Java can be expected to go, and soon I was satisfied with the first two dialog pages and ready to hit the third. For the UI, I went back into Visual Café and started designing the form. I like Visual Café, but as with any development tool, a bug in the tool can make for a painful hour or two for the user. It turns out that Visual Café does not handle the backslash in label names correctly; if a backslash is the last character of a label, the effect is a C programmer’s nightmare: lines of code become literals and literals become lines of code (the closing quotes becomes a ”, or literal quotes, instead). Unfortunately, since Visual Café reads and interprets the source without ever compiling it, the result is a very flaky user interface accompanied by much lost data and gnashing of teeth. Two hours later I had diagnosed and fixed the problem, so I finished the UI and got the heck back to Visual J++.

Before packing it in for the evening, I found a serious problem with what I believed at the time to be the IE3 Java virtual machine (VM). For some reason, the String::substring function was failing in a
new and spectacular way. Using the debugger, it became clear that there was a problem internally with the function call mechanism and that I would either have to get a fix from MS or write the function myself. Thus disheartened, I retired for the evening to drown my sorrows in a couple of NBA playoff games.

Day 4: 5 hours

**Day 5: Virtual Success!**

As luck would have it, my inbox held an e-mail surprise for me. It turns out that Microsoft’s Java VM has a few problems, and they had made a patch available on the web (they were kind enough to point out that Sun’s VM had a few more problems than theirs). “Hooray!” thought I, as I fired up the browser. Anticipation turned to anxiety which was followed shortly after by boredom on the heels of disgust — Microsoft’s web site was maxed out and completely unreachable. “Yikes!” thought I, as I began to work on a workaround for yesterday’s substring() problem.

After playing with a few UI optimizations such as synchronizing the header and message pages, I fired up the applet and decided that I had a working prototype. The next step: the web.

Day 5: 4.5 hours

Complete prototype time: 20.5 hours

**Day 6: All Things in Moderation...**

After kicking off the afternoon by finding a couple more bugs in Visual Café and the Symantec classes, I began to write the moderator program in VB5. For this I decided to use the Smithware ActiveX Controls client for the IDS, so that we could moderate postings via remote site or over the LAN. Although I’m not thrilled with the idea of making the DDF’s publicly available, the owner name should give adequate protection, at least for this low-security applet. Were security a larger issue, I would have enabled security on the IDS itself.

Writing the VB5 application was pretty straightforward, except I realized another unforeseen design flaw — the files did not allow the moderator program to “skip” postings effectively. Most of the afternoon was spent changing the file layout, recreating the files, changing the applet to reflect the new file layout, and testing the data. Nothing I love better than re-doing work for a one-byte field change!

Day 6: 5 hours

**Day 7: The Thrill and the Agony!**

If you’ve ever read the book *Alexander and the no-good, very bad, terrible day* then you can just skip the rest of this section. Here’s an adage that Benjamin Franklin left out: “Long weekends do not make for profitable Tuesdays.” After spending most of the day accidentally overwriting much of Friday’s work, I finally managed to get the applet on the web site and lo and behold, the clouds parted, a ray of light shone down, and I beheld the face of Java.

Actually, it rained all day and the only face of note I saw was Boris Yeltsin’s on a news site, but the effect was equally dramatic — the applet worked. I was even able to download it over my 14.4 modem and post a message. Although the test web page containing

---

**Btrieve Explorer™**

**SMARter, **Simpler, **Faster... Better!**

...by Btrieve users... for Btrieve users!

**EXPLORE YOUR DATA IN A LIGHTNING FAST VIEWER DESIGNED SPECIFICALLY FOR BTRIEVE...**

...by Btrieve users... for Btrieve users!

**SMARter**

Automatically preserves useful user settings such as field layout and sort.

**Simpler**

Allows simultaneous viewing of any number of data windows.

**Faster**

Obtains maximum performance via 32-bit, multithreaded processing and direct MicroKernel access.

**FREE DOWNLOAD!**

See IDS, page 17
Going Public:
Connecting Your Database With The World Wide Web

By Howard Shirle
choiceword@aol.com

You knew it was going to happen the moment you looked at a soda can and saw www.drpepper.com. Some day someone was going to walk into your office and say, “So, how do we put our corporate data on the Internet?” Your initial response was probably a silent scream of “We don’t!” And who could blame you? The Internet can be a scary place for data; it’s hard enough keeping it correct and secure within your own corporate network.

But the Internet can offer advantages for many business that may outweigh the risks. The greatest advantage is simply universal access. If you have far-flung offices or personnel on the road, or if you simply need to give customers access to information, the Internet offers an easy route to your corporate data from almost any location.

The drawbacks are mostly obvious, of course. Allowing Internet access to your database increases the opportunities for a malicious hacker or a corporate spy to damage or steal your data. Nor is the Internet the most reliable connection; your users may find their service interrupted for a host of reasons, most completely beyond your control.

But you’ve considered all that, and decided the pluses outweigh the minuses. You’re ready to take the big leap. The only question is, which way do you jump? How do you connect your delicate data with the Internet beast and produce a marriage everyone can be happy with?

The Common Gateway Interface

The Common Gateway Interface, or CGI, was developed to provide a cross-platform standard to allow web clients to communicate dynamically with a server. At the moment, CGI is the primary tool for connecting a browser to a back-end database. CGI allows the client to activate a program on the server which interprets the client’s requests and returns the appropriate result as a new HTML page. The advantages of CGI are clear: First, it is a recognized standard method which all browsers can use. Second, since a CGI application runs entirely on the server, compatibility problems are essentially eliminated; the client can be any browser on any machine running any operating system, and nothing needs to be downloaded to the client except for the original HTML document and the server-generated results. CGI applications can be developed in almost any common programming or scripting language, although C and Perl are generally used the most.

The drawbacks of CGI, unfortunately, are directly linked to its advantages. Because the CGI application is resident on the server, the server must handle all of the processing tasks beyond the simple display of the HTML Web page. And because the CGI application is just that, an independent application, it launches and closes every time it is called. This can be a considerable waste of time and a drain on server resources, especially if the server is receiving multiple CGI calls from separate clients. With each call, the CGI application launches, loads into memory, executes its commands, closes itself, and launches itself again for the next call. Imagine your users having to quit and restart your database search engine every time they wanted to conduct a single query! That’s an exaggeration to be sure, but the performance hit of CGI can be very real. And when you consider that multiple versions of the same CGI application are opening and running as each client makes a request, how slow can you make your server go?

Also, CGI is stateless; once the request is made and the results are returned, the server maintains no connection with the client. Subsequent changes in the database will be invisible to the client unless it initiates another CGI request. And this stateless design means CGI doesn’t store information about the transaction, including whether it was successful or if any errors occurred; imagine an application that can alter your data but not confirm that it did so correctly and therefore can’t abort or undo the transaction. And since CGI manages each transaction separately, two or more users could attempt simultaneous changes to your data and never know that their requests conflict. The “stateless operations” drawback can be overcome; you can, for example, write code to handle these problems using “cookies” or other solutions. One solution for the weaknesses of CGI is offered by Open Market Corp. in Cambridge, Mass. Called FastCGI, this alternative set of CGI extensions seeks to alleviate both the performance hits and integrity problems of CGI.

FastCGI works by keeping the CGI application open on the server as a separate process. This eliminates the slow down associated with repeatedly opening and closing the CGI application. Fast CGI also offers solutions for the transaction problems through authentication and authorization checks.

Unfortunately, FastCGI is not yet supported by all Web servers. Most notably absent are the servers developed by Microsoft and Netscape, the two biggest bulls in the arena.

Web Server APIs

So what do Microsoft and Netscape propose? As is to be expected, something completely different especially from each other. They each propose APIs (proprietary to their respective servers, naturally) which allow programs to access the server directly: Internet Server API for Microsoft and Netscape Server API for Netscape. While at first blush this doesn’t seem much different from CGI, the advantages
come in the differences between the two methods. Where CGI calls and executes a program (and as we’ve seen, a separate one for each client), APIs use routines which are directly available to the system, can be called as needed by the server, and can remain in the memory instead of automatically closing when each task is finished. The APIs also offer the advantages of multithreading, allowing several clients to use the same routine without forcing the server to run multiple simultaneous copies.

Although the APIs are available publicly and many tool vendors and database engine suppliers are developing links for these servers, these are strictly proprietary tools, and neither Microsoft nor Netscape appear likely to accept the other’s “standards” as their own. Also, so far these APIs do not offer session and transaction management operations, raising most of the same problems as CGI. And since the APIs use the same address space as the web server, a flaw in your code could hit you with a nasty crash; the last thing you want when both your data and your web site are on the line.

Microsoft’s Internet Database Connector (IDC), part of Microsoft’s Internet Information Server (IIS), is another server-based method for linking your database to the Web. With IDC, the browser simply loads a script located on your server. This simple text script holds SQL statements (which can be as elaborate as needed, restringed only by the capabilities of your database engine and ODBC driver) and identifies a separate HTML template you have defined on the server for it to use to build the response page. The server processes the results of the SQL query, plugs the answers into the HTML template, and returns the page to the browser. These are not the limits of an IDC script; IDC supports a number of parameters, variables, and conditional operators, so you can give your users a number of ways to interact with the data. Drawbacks are IDC’s proprietary nature — it’s only available with IIS — and the limits mentioned earlier with respect to other HTML data publishing solutions. IDC is also not as powerful as some other options, and the processing tasks are placed almost exclusively on your server.

Microsoft’s follow-up to IDC is dbWeb, which also comes as part of IIS. dbWeb uses ODBC drivers to access your database, calling on schema you set up to determine the details of the query. dbWeb then returns the data as an HTML page. The schema gives dbWeb its strength. Using the Schema Wizard, you can easily create a customized query system with surprising power. One big plus is the ability to add drill-down fields to the result page. This turns the returned data into hot-links; users can click on a response and move further down through the data. dbWeb also lets you create several query methods for users, and even bypass the wizard to create even further customized schema and pages.

Like IDC, however, dbWeb is exclusively part of Microsoft’s IIS package. Nor does it give you quite the level of control or processing power which a more direct link to your server offers.

Web Development Tools

Other profit-minded companies are offering connection solutions and development environments designed to address the shortcomings of CGI and Web server APIs. These solutions take many forms, from scripting languages to propriety extensions of the HTML tag set to full-fledged visual development environments similar to Visual Basic and Delphi.

If you haven’t encountered it already, HTML (HyperText Markup Language) is based on a system of formatting tags included...
in text files which browsers interpret to produce everything from simple highlighted type to input forms. Several vendors offer solutions based on extensions to the HTML tag set aimed at the server, not the browser (browsers are supposed to simply ignore tags they cannot interpret). You place these tags in your HTML text file when you create a web page. When a browser sends a request, a proprietary server application (running in addition to your web and database servers) interprets these tags and interacts with your server accordingly. This method allows you to turn the client’s browser into a customized interface without having to resort to coding. You do have to be familiar with HTML and tags, but that is considerably simpler than attempting the same results in C++.

Other products take their cues from visual programming languages. Rather than ask you to learn a scripting language or tags, these products allow you to drag and drop objects to build fully coded custom applications. Your programs are compiled and run on your web server, interpreting browser requests and publishing dynamic HTML documents containing the data.

All of these products are essentially additions to HTML and HTTP, using the CGI, an API, or both. The disadvantage to this is that HTML is well suited for data publishing but not for data entry; it just wasn’t designed to work that way. Another drawback is that the processing demand falls almost exclusively on the server. Interpretation, parsing, and validation must all occur on the server, when these tasks could be easily distributed to the client. While this server-side architecture does assure almost guaranteed compatibility with any client platform, it places undue demands on server resources.

Recent technological advances make it possible to ease that demand and bypass HTML almost entirely. With Java and ActiveX, you can pass more of the processing control to the client. Under these technologies the browser could act simply as a container for applets (small Java applications) or ActiveX objects. These, in turn, would run on the client machine, providing for a more powerful, customized user interface and added functionality not possible with straight HTML.

But Java and ActiveX are a programming language and a component architecture, respectively, not Web-based development environments. Sun Microsystems has only recently proposed JDBC (Java DataBase Connectivity) drivers, of which only a handful are available. (Java can work with ODBC drivers.) Fortunately tools are available to help you develop Java applets and ActiveX objects for Internet-to-database applications; some of the CGI/API development tools do double duty on this front.

In the midst of all these gee-whiz-bang, do-it-all-for-you tools, Smithware’s Btrieve-based Internet Data Server (IDS) deserves special notice. By far most of the developer’s tools available today are targeted towards Web browsers as the client’s primary interface (In my search, all the tools I found were designed with browsers in mind). But the Internet is not the Web, and browsers aren’t the only game in town. The Smithware ActiveX Internet Data Server and its DLL, ActiveX control, and Java clients provide the simplest and most direct access to data possible without skimping anything in power. No need for a browser, no need for HTML tags, CGI, HTTP or any other acronym. Rather than being limited to the Web, Smithware’s IDS simply uses the Internet as a communications pathway for a live database connection.

I went to their office to see for myself. I sat down with Ken White, IDS’s lead developer, for a demonstration. Ken led me through the steps of building a search application in Visual Basic using the client ActiveX control. It took all of 30 seconds. He pulled up a form, added the control, gave the application an IP address, drew a couple of fields, set their properties, and there we were, accessing and changing live data over the Internet. Then he did it again in Delphi. He’d have probably done it again in Optima++ if I hadn’t looked so stunned.

Instead he took pity on me and introduced me to the server side of the package. Full Windows NT Domain security. Multithreaded. Selectable data compression (the server can also sense whether a client is using compression and respond in kind). Should your Internet connection fail, IDS is smart enough to start up at the break when the connection is re-established. IDS also checks for lost clients, so that resources aren’t tied up by broken connections (or absent-minded users). And the server can handle multiple clients from multiple locations over multiple types of connections, all simultaneously. On the client side, a single user can have as many open connections as his system memory will allow. And once you give the client an IP address for a server, it knows where to look for that server, regardless of what machine the client app is on. No location specific coding is necessary. Best of all, Btrieve does not have to be installed on any client machine, only on the server.

Just think about it. In a couple of hours you could develop a robust, secure application that anyone (or just certain people, if that’s what you need) can use to interact with your database from practically any location. You can plug the client into a web page for browser access if you need to, but if you don’t need to, you don’t have to.

And the drawbacks? Currently the server process is implemented only on Microsoft Windows NT Server, and Ken wasn’t 100 percent happy with the access delay. How much of a delay? Blink. Sorry, that was too long. So Ken’s 99.9 percent happy. Something tells me you won’t quibble over it.

---

**KnowWare, Inc.**

Software Systems GmbH

- Technical Support
- Inhouse-Service
- Sales
- Programming
- Tools

**B-STRESS**


$799

**B-MIRROR**

Mirroring Btrieve files at runtime: without stopping Btrieve, improved security, approaching replication.

(on request)

**B-ROLL**

Advanced Rollforward for Windows: Logfile-Tracing, stop at a selected operation.

$239

---

**KnowWare Software-Systeme GmbH**

Steubenstrasse 9  D-33609 Bielefeld

Tel: (+49) 521 93259 0  Fax: (+49) 521 93259 11

http://members.aol.com/knowwarede
Btrieve and NDS Configurations

Btrieve for NetWare, v6.15

ISSUE: When users are logged in to a NetWare 4.1x server via NetWare Directory Services (NDS), a Status Code 94, “Permission Error,” or Status Code 20, “Btrieve Not Loaded,” may be returned when trying to access Btrieve files located on the server. Most of these problems occur when running a 16-bit or 32-bit Btrieve application in conjunction with the 32-bit Btrieve requester.


You can also download the files BTRV&NDS.DOC from the Pervasive FTP site (ftp://ftp.pervasive-sw.com/support/refshelf/nds.doc) and CompuServe (GO BTRIEVE) in Library 4 as BTRV&NDS.DOC.

This information is continually updated as new processes or fixes are discovered.

Following are various troubleshooting steps that may resolve this problem. All information relating to the setting of the Bindery context is per Novell’s documentation and is not specific to Btrieve.

1: The bindery context on the server where Btrieve is running MUST have a Read/Write replica of the partition of the container object for which the bindery context is set.

2: The user (or alias) MUST exist in the container object for which the bindery context is set.

3: Creating a Btrieve object using the Btrieve Setup Utility (BSETUP.NLM), which adds a LOAD BDIRECT line to the BSTART.NCF, solved one customer’s Status Code 94 problem.

4: When running a 16-bit application on a Windows 95 or Windows NT workstation, the recommended configuration is to set the following entries in BTI.INI:

   [Btrieve]
   local=no
   requester=no
   thunk=yes

   With this configuration, the 32-bit requester is used to communicate with the server. It is sometimes helpful to know if the problem occurs with only the 32-bit requester, or with both the 16-bit and 32-bit requesters. In this case, alter the BTI.INI to have:

   [Btrieve]
   local=no
   requester=yes
   thunk=no

   If the 16-bit requester works and the 32-bit requester does not, the problem may be one of the following:

   a: The Btrieve object;
   b: Mismatched networking DLLs; some customers have replaced the MS NWNP32.DLL and resolved the problem;
   c: Username/password problem - make sure the user name used to log into the Windows 95 or Windows NT workstation is the same as the username used to log into the server; or
   d: The 32-bit requester v6.15.440 recently released by Pervasive may have a problem with NDS. It may be necessary to use the previous (6.15.435) release to resolve this problem.

5: If both the 16-bit and 32-bit requesters return an error, the problem may be one of the following:

   a: The bindery context on the server is set wrong.
   b: The Windows NT or Windows 95 user logging in to NDS is configured with the MS Client for NetWare instead of the Novell Client (refer to the NDS Document for more information about this.).

Support of DOS applications on NT clients

Btrieve for DOS Workstation Engine and Requesters, v6.x

ISSUE: A Windows NT workstation is a supported platform for DOS applications. Both Btrieve for DOS v6.15.x and Btrieve DOS requesters (BREQNT.EXE or BREQUEST.EXE) targeting a Windows NT or NetWare server running a Btrieve engine can be used to run a DOS Btrieve application. However, some software vendors may not be compatible with Windows NT DOS boxes, and therefore, they may not recommend that their software run in this environment.

If a user needs to know whether a specific software package is supported in a specific environment, the user needs to consult with the application vendor.

Status Code 20 When Running Application on Windows NT 4.0 Server

Btrieve for NT

ISSUE: When trying to run a 16-bit or 32-bit Windows application on a Windows NT 4.0 server running the Btrieve for NT Server Engine, the application could receive a Status Code 20, “Btrieve Not Loaded.”

SOLUTION: Copy WBTRV32.DLL from the \BTI\WinNT\Bin directory to the Wint40\system32 directory. This will enable the MKDE Interface option in the Btrieve Setup Utility (W32MKSET.EXE) drop down box. Select this option and set Target Engine=Server. Stop and start the Btrieve Service again, and then retry the application.

GPF from 16-bit Application with 32-bit TCP/IP Requesters

Btrieve for Windows NT, v6.15.435

ISSUE: A 16-bit Windows application caused a GPF when opening a Btrieve file on the
Windows NT server running Btrieve for Windows NT v6.15.435. Verified that the correct Btrieve requesters were being used, and TCP/IP was working with ping.

**SOLUTION:** The registry did not show the Communication Requester entry under HKEY_LOCAL_MACHINE\SOFTWARE\Btrieve Technologies. Once the REGUPD.EXE utility provided with the 6.15.435 release ran, the registry entries were created. After rebooting the workstation, the requesters worked correctly. Registry entries will also be written correctly if the Btrieve Function Executor is used to open a file using the requesters.

**Status Code 3 After Many Inserts Using TCP/IP**
Btrieve for Windows NT, v6.15.435

**ISSUE:** After an application inserts more than 90,000 records, a Status Code 3, “File Not Open,” is returned. Changing Btrieve parameters such as bundle limit did not affect the timing of Status Code 3. This problem occurred at a customer site, but the same application ran successfully at other sites.

**SOLUTION:** Update the TCP/IP protocol stack. The customer had the original Microsoft TCP/IP 32 protocol. After installing version B, the problem was solved.

---

**Q & A**

**Q:** I tried to reinstall Scalable SQL 4 on a server and was returned a Status Code 7109 (“Serial number sync error’). Why?

**A:** You cannot install over an existing configuration without first backing up and removing some key files. The INSTALL.TXT lists the four files (they are different for Windows NT and NetWare) that the serial number is written to. The most common problem seems to be caused by NWUCMGR.SYS for NetWare users. The NLMs related to Scalable SQL need to be removed as well as NWUCMGR.SYS. Because the date on the NWUCMGR.SYS is newer than the install version, it is not overwritten during a subsequent install. So the NWUCMGR.SYS file ends up having a different serial number (serial numbers are randomly generated during install) than the other three files the number is written to. This results in a serial number synchronization error.

**Q:** I am getting unusual results from my Scalable SQL or ODBC queries. How can I verify if my Data Dictionary Files (DDFs) are created correctly and valid?

**A:** When using Scalable SQL or ODBC and seeing unusual results in queries occurring (i.e., records displaying multiple times, sorts not working, etc) the keys in the DDFs do not match the keys in the data file. This can be verified by running BUTIL -STAT or using the Btrieve File Manager on the data file. Use SQLScope, DDF Builder, or in ODBC cases, do SELECT * FROM X$Index and SELECT * FROM X$File to view the DDF information. The ODBC cases are the hardest to verify because the files are referenced by a number in INDEX.DDF. Query FILE.DDF (X$File) to see what number goes with which data file. Also, the DDF's themselves may be defined incorrectly. Use BUTIL-STAT to compare your DDFs to the DEMO DATA DDF. Except for the number of records, the information should match. For example, if the DEMO DATA FIELD.DDF statistics show a case insensitive attribute on key 2, the DDF in question should show the same. Some of the more common errors generated as a result of inconsistent DDFs are 238, 204, and 207. Just about every Status Code related to fields or indexes. Status Code 238 is by far the most common. This is caused when the indexes in the DDF do not match the data files indexes. This can be caused by key lengths not matching, ACSVs not matching, or just about any other key attribute out of sync between the data and DDFs.

**Q:** Why is a Status Code 2306 (“Invalid UNC path”) returned when adding database names under Scalable SQL 4?

**A:** With Scalable SQL 4, make sure DBNAMES.CFG is in version 6.0 or 7.0 file formats. If the file is 5.x format, then a Status Code 2306 (“Invalid UNC path”) is returned when adding a named database. If tracing is turned “on” under the Btrieve MicroKemel Engine, a Status Code 84 is logged on the Insert to DBNAMES.CFG. Also, be aware that trying to access a 7.x file version formats with a lower engine version (i.e., Btrieve v6.15), returns a Status Code 30. “Not a Btrieve file”

**Q:** Why do I receive a Delphi code “invalid parameter query” error when attempting to use ?? (then) operator in a SQL statement?

**A:** These scalable SQL functions do not work in Delphi. Delphi treats the ?? (then) variables. A substitution can be made to use ‘then’ and ‘else’ in the code, and still obtain the same results. This is not a bug in Pervasive’s product; it is an interoperability issue with Delphi. A Delphi ‘invalid parameter in query’ error is returned when attempting to use these functions in Delphi.

**Q:** When running SQLScope or a 16-bit Scalable SQL or ODBC application on a Windows NT/95 workstation targeting Scalable SQL 4 on a server (Windows NT or NetWare), strange unexpected errors are returned, or there is no data returned when there should be.

**A:** This could be due to the communications buffer being too small, and a Status Code 2106 or something similar should be returned. Increase the communications buffer in three places:

1) 16-bit setup, SQL Requester/Access Control/Comm Buffer Size.
2) 32-bit setup, SQL Requester/Access Control/Comm Buffer Size.
3) Either setup, Connect to server, Scalable SQL Communications Manager/Server Communication Configuration/Comm Buffer Size

(Do not forget to unload and reload Scalable SQL at the server.)

Changing the server is the most important; the requester communications buffers are dynamic and increase when needed, so they should not have to be increased unless you want the larger buffer to be initiated at requester load time. This problem may be more apparent when not thunking, however, there have been cases where thunking was set to “on”.

**Q:** Once you introduce ODBC into a Btrieve environment (or Scalable SQL for that matter), how do you protect or secure the data from the users? For instance, in an accounting application, how do you keep users from getting to payroll data once they have ODBC capability? Is there encryption in Btrieve?

**A:** Yes, encryption is available in conjunction with owner names. However, if you put owner names on Btrieve files, the Btrieve application will not be able to access them either.

Security is supported in the Scalable SQL level through database security. You can install security on DDFs, assign user names and user groups, and assign field and/or table rights for read and write access to each user or group. Because this is all managed via the DDFs, Pervasive's ODBC driver will respect this security.

**Q:** Does Pervasive Software support the local engine for Scalable SQL (16-bit) under Windows NT?

**A:** The local Scalable SQL Client Engine for Windows is supported under Windows NT/95. Of course, this comes with the Btrieve Client Engine for Windows, which is also supported. However, this configuration only provides support for 16-bit ODBC drivers. Running 32-bit ODBC applications, requires Btrieve for Windows NT/95, or Scalable SQL for NetWare or Windows NT Server Engine.
Q: What can I check when using the ODBC Interface and database opens seem to be delayed?

A: Check whether the trace is set to "yes" in the ODBC administrator, then check the registry in Windows 95 machine, and change trace=0 under the registry key: H_KEY_LOCAL_MACHINE/software/odbc/odbc.ini/odbc.trace set to 0.

Q: Why can I not see the named database list in SQL scope when using Microsoft Client on a Windows NT or Windows 95 workstation?

A: Named databases are only broadcast on the server. Reboot the workstation, then run the ODBC administrator, then check the registry key: H_KEY_LOCAL_MACHINE/software/odbc/odbc.ini/odbc.trace set to 0.

Q: Why can I not see the named database list in SQ L scope when using Microsoft Client on a Windows NT or Windows 95 workstation?

A: Named databases are only broadcast on the server. Reboot the workstation, then run the ODBC administrator, then check the registry key: H_KEY_LOCAL_MACHINE/software/odbc/odbc.ini/odbc.trace set to 0.

Q: How can I isolate a problem where Btrieve is returning a Status Code 20 from the workstation and multiple network interface cards are in the server?

A: Try the following. Connect the workstation into Net1 and disconnect the Net2 NIC in the server. Reboot the workstation, then run the application to test Btrieve. Note whether a status 20 was returned or if the application was successfully run. Next, disconnect Net1's NIC and connect the workstation into Net2. Reboot the workstation then run the application to test Btrieve. Note whether a status 20 was returned or if the application was successfully run. If the Btrieve tests were successful, this would indicate a network routing problem. Advise the network administrator to check routing configurations for the particular network.

Q: Are there any compatibility issues with Btrieve 5.10 and Windows 95?

A: Btrieve for DOS and Btrieve for Windows v5.x has never been tested on Windows 95. We do not support this configuration. Our shipping products at the time Windows 95 was released were Btrieve for DOS and Btrieve for Windows v6.15. There are many customers who use Btrieve for DOS v5.10a under Windows 95 without experiencing any problems.

Q: What is the advice on what Btrieve package to use when migrating from Btrieve v5.x to Btrieve v6.15. One solution fitting four configurations which are as follows:

1) Windows 95 client requesting data from a NetWare server
2) Windows 3.11 client requesting data from a NetWare server
3) Windows 95 client requesting data from a peer-to-peer to a Windows 95 server
4) Windows 95 client requesting data from a Windows NT server

What product is best going to solve all these in one option?

A: Since the environments are client/server and peer to peer and the solution is to have one product, the choice must be a client engine. However, because the clients are Windows 95 and Windows 3.1x, use Btrieve v6.15 workstation engine for Windows 95/Windows NT in peer-to-peer and on NetWare and NT servers. Use Btrieve for Windows v6.15 Windows workstation engine on peer-to-peer and on NetWare and Windows NT servers.

Summary:
Purchase needs are workstation engine versions of Btrieve for Windows NT/Windows 95 and Btrieve for Windows. If there were no peer-to-peer needs, then Btrieve for NetWare and Btrieve for Windows NT server engines would be the best choice.
However, always keep in mind the most important question to ask — what kind of application is running? If it is a DOS application, it does not matter if the client is Windows 3.1, Windows 95, Windows NT, or OS/2. The need is for a Btrieve for DOS workstation engine. If it is a 16-bit Windows application, then the summary stated above is correct.
If running in a Windows NT or NetWare server environment, then this question is not important, because the client/server engines support both types of applications.

Q: Does the Btrieve for DOS developer kit allow you to modify functions that make up a library that gets linked with the executables? Would a file that is in Btrieve 5.x need to be modified to work in protected mode?

A: You cannot modify the functions in the interface libraries. Our real-mode DOS interfaces do not work with protected mode compilers, with the exception of Borland PowerPack (C/C++), PharLap v6, and Borland Pascal v7 protected mode. The other supported environment that allow you to develop a protected mode DOS application is the "DOS/4G for Btrieve" package from Tenbery. A Btrieve file NEVER changes its structure relative to the application that is accessing it. A Btrieve file is a generic cross-platform file. The same Btrieve file is used by a DOS real-mode, DOS protected mode, Windows, or OS/2 application simultaneously. There are only two cases where this is not true: 1) you cannot have an application (running on any platform) that uses a workstation engine access a Btrieve file at the same time an application (running on any platform) is accessing the file using a client/server engine. 2) If a Btrieve file is a particular version, an engine (workstation or server) that is of an earlier version cannot access that file; for example, a Btrieve for DOS v5.10 engine can not access a 6.x Btrieve file.

Q: Does Btrieve support on-demand disk flushing?

A: You can tweak some parameters that control when Btrieve writes out its cache (g), but on demand is not available. You can refer to ftp://ftp.pervasive-sw.com/bin/refshelf/WEPERF.DOC on the FTP site for additional information.

Q: Does Btrieve support custom collation for sorts?

A: Btrieve supports alternate collating sequences on string, IString, and zString keys which allow a user-defined sort to determine the ordering of the data.

Q: What version of Btrieve for Windows NT server supports the TCP/IP requesters (Windows, Windows 95, Windows NT)?

A: Btrieve for Windows NT Server v6.15.1935 or later.

Q: I understand that the NetWare version of Btrieve supports SPX. What about SPX II, which is able to handle packets larger than 576 bytes. Does Btrieve support SPX and SPX II?

A: Pervasive does not support SPX II.

Q: Can Btrieve read the fixed-length portions of a variable length record, by setting the data buffer size to the fixed-length and then update, the record without losing the variable portion?

A: The fixed length portion can be read by setting the data buffer size as described; a Status Code 22 and the fixed portion will be returned. However, an update will lose any variable portion associated with the record. Instead, perform an Update Chunk to only update the bytes starting at offset 0 for a length of <length of fixed portion>. This option will retain the variable portion.

Q: Does Pervasive support Btrieve for Windows (16-bit) engines under Windows 95? If so, does Pervasive Software support the Btrieve Windows Engine (16-bit) under Windows NT?

A: Pervasive Software supports the Btrieve Client Engine for Windows on Windows 95 and Windows NT. If someone has a problem with this configuration, we need a test case to demonstrate it. The preferred method of running a 16-bit application in a Windows 95 or Windows NT environment is to use the Btrieve for Windows NT/Windows 95 workstation engine.
n ongoing effort to provide the best service possible to its customers, Pervasive Software has recently combined its customer support and maintenance programming areas to create a Customer Engineering group.

According to Pervasive executives, maintenance programmers working directly with software support engineers (SSEs) provide a winning combination. Engineers, whose main responsibility is to support Pervasive’s customers, will be immediately available to provide in-depth diagnostic help to Pervasive’s software support engineers. This new structure enables the product development organization, the group responsible for the development of new products, to stay focused on hitting their release targets. From an SSE’s perspective, this new environment provides the opportunity to work with maintenance programmers and hone their diagnostic skills while also providing additional career growth opportunities. But the real winners are Pervasive’s customers. Who will receive needed engineering support in real time.

In conjunction with this restructuring, Pervasive is also forming a training group, based at its headquarters in Austin, Texas. This group will teach in-depth classes in Btrieve and Scalable SQL, and will train its Channel and Manufacturing partners to provide front-line support to their respective customer base. A Pervasive consulting group is also being planned for January 1998 to provide on-site solutions and support without conflicting with the goals of Pervasive’s Channel partners.

The formation of the Customer Engineering organization is just the most recent change designed to bring improved service and support to the company’s manufacturing partners, developers, channel partners, and end users. Over the course of the last 12 months, there have been significant changes contributing to the improvements in support.

Real-time Support

In February of this year, Pervasive’s customer support group took a major step in its continuing effort to provide quality support for its developers, customers and other end users by implementing a real-time support strategy. Conversion to real-time support provides direct access to software support engineers, thus eliminating call back delays and the never-ending game of telephone tag. No longer do callers to Pervasive’s Austin-based support center have to wait for a return call to receive the help they need. In the real-time support model, qualified SSEs are available to begin immediately diagnosis of problems.

Pervasive’s telephone support center consists of a front-line and a back-line support group. The Front-line group is staffed by 12 SSEs providing coverage from 7 a.m. to 7 p.m., Monday through Friday. Since its inception in February, the Front-line group has consistently handled more than 90 percent of the incoming telephone calls in real time. This is a significant improvement over the Dispatch/Call Back model previously employed by the group.

In the previous support model, calls were handled by non-technical dispatchers who entered the request for support into the call tracking system. Software support engineers would then attempt to return the call and, while their first attempt to return the call occurred within 30 minutes, the ensuing game of telephone tag often delayed delivery of support for hours. In the new business model, telephone hold times have averaged from two to four minutes before callers gain access to front-line staff. Pervasive is working hard to recruit and train additional Front-line staff with the goal of reducing telephone hold times to 30 seconds or less.

The Front-line group is staffed with a mixture of seasoned Btrieve veterans and new SSEs who have gone through Pervasive’s intensive new hire training program. This group handles the majority of incoming telephone calls and electronic requests for support. New incidents requiring escalation are passed to the Back-line group. The seven-person Back-line group, managed by Linda Anderson, provides technical support to the Front-line and they also handle incidents escalated from the Front-line.

Additional back-line responsibilities include account management for Pervasive’s manufacturing partners and participation in product planning, beta testing and new product roll out.

Over the last 12 months, there has been a significant improvement in the level of support and the options available to Pervasive’s customers. In the first calendar quarter of 1996, customer support faced a large increase in call volume and an enormous backlog of open calls that were fueled by the dramatic increase in sales of Btrieve client/server, workstation engines, and developer kits. During that period of time, Pervasive’s technical support group virtually tried to catch up while recruiting and training efforts were accelerated.

Premium Support Program

In May 1996, Pervasive introduced its premium support program. The introduction of a fee-based program significantly decreased daily call volumes and allowed Pervasive to focus its efforts on its developers, VARs, and corporate end users are the people who most needed and deserved their attention. The decreased call volume, along with new additions to the staff, led to a marked improvement in the delivery of support services. It took a few months to work through the large backlog of calls, but Pervasive was definitely on the right track, and the group’s metrics steadily improved. Support was now acceptable, but Pervasive knew that it could not achieve success without a transition to a real-time support model.

Customer Support on the Web

Another major milestone was the introduction of the Pervasive support web site in July 1996. The support web site contains technical papers, tech tips, and other information Pervasive hopes its customers find useful. Pervasive’s intent, as you would expect, is to share as much information as it can to help its customers and users solve problems on their own and not have to call Pervasive at all are a win/win strategy. In the next few months, Pervasive will be improving its web site with the addition of a password-coded section for its business and manufacturing partners and contact customers. Pervasive will also be offering improved search capability to easily locate needed information. There will be a major, company-wide effort to make relevant information available through the web site.

Pervasive Tech Talk

In August of 1996, Pervasive introduced Pervasive Tech Talk, a bimonthly publication sent via e-mail to its manufacturing partners and contract customers. This valuable publication is chock full of up-to-the-minute information on new software releases, newly discovered software defects, and the latest tech tips. Tech Talk has drawn rave reviews and certainly adds value to Pervasive’s premium support offerings, and duplicates the information offered in the Technical Corner of BDJ.

Pervasive European Support

In September 1996, Pervasive opened its European support center in Paris. In February 1997, the support operation relocated to the company’s new Dublin support center. The Dublin facility, headed by Alex McDonnell, will provide pre- and post-sales support to Pervasive’s European developers, distributors, and other users of Pervasive products. The Dublin office will also house an engineering staff to localize products for the European markets and provide specialized post-sales engineering and consulting.

Customers requested the enhancements described in this article. Pervasive Software will continue to listen to its customers and will try its best to deliver on their expectations. This latest change is a great example of Pervasive’s commitment to continuous improvement.
New Btrieve Newsgroup Available on Usenet

By Kathy Jones
kathy@smithware.com

It’s been in the works for months, and now it’s official! Btrieve has its own unmoderated newsgroup: comp.databases.btrieve.

There are thousands of newgroups available, and some of them, like comp.databases and comp.databases.ms-access, even discuss Btrieve. But until now, the Btrieve Forum on CompuServe has been the only place where programmers and developer’s could ask specific questions or share their experiences using Btrieve. But what do you do if you don’t have CompuServe? Now, you can visit the comp.databases.btrieve newsgroup (Not to worry! CompuServe subscribers are welcome to visit too!).

If you are already using newsgroups, simply use your news reader software to subscribe. If comp.databases.btrieve does not appear in your news reader’s list you may need to contact your Internet service provider and ask them to add it to their news server.

If you have never used newsgroups before, the easiest way I’ve found to connect to comp.databases.btrieve is to visit the DejaNews web site at http://www.dejanews.com, the self-proclaimed “Source for Internet Newsgroups.” Or you may want to obtain and install news reader software. Microsoft Internet Explorer 3.0, for example, includes a news reader option. Contact your ISP to find out how to connect to their news server. For more information on newsgroups in general and news reader software, see Doug Reilly’s story, Getting the News on page 18 of this issue.

If you have a question about Btrieve, please post it. I use comp.databases.btrieve for BDJ story ideas!

IDS ...continued from page 8

the applet was spartan, the applet was still a sizable download, so I decided to compare the access speed of it to that of other sites. On a 14.4 modem, I was able to retrieve the applet in a little over a minute. After several failed attempts to connect to the Microsoft web site (apparently it was not really where I wanted to go today), I decided to compare the applet site to one of the few vitally important sites on the entire Internet — ESPNet SportsZone http://www.espnnet.sportszone.com. The home page of the ‘Zone contains an applet and several pictures; all told, the entire page took over two minutes to come across, for a total download time about 250 percent greater than that of the discussion applet. “Satisfactory,” I decided.

That night, I tested the applet from home and to my amazement, the thing worked.

Day 7: 5 hours

All told, the entire process has thus far taken about 30 hours. There are still a few small things left to do (mostly UI improvements), and there needs to be an HTML page around the applet that describes it, but all in all, the applet works. Not bad for less than a week of work.

Now, most faithful BDJ readers who have made it this far are beginning to feel as if...well, as if something was missing. “What is it?” I can almost hear them say, scratching their heads. Here it is. So far, this article has had very little to do with Btrieve. (“Oh, yeah — that was it!” say the BDJ faithful.) If this article is about using Btrieve in Java, then why haven’t I mentioned Btrieve yet? Except for a few passing references to extended operations, this might as well have been a Java article. What’s up?

The answer to that is very simple: Btrieve was the easiest single

See IDS, page 20

Southdown House Software Limited
Providing the following specialist Btrieve services in the U.K.

• Technical Support  • Consultancy
• Analysis  • Installation
• Optimisation  • Training

Authorized distributors for

Smithware, Inc.

Providing the following specialist Btrieve services in the U.K.

Tel: (01306) 877998  Fax: (01306) 887755
Internet: info@southdown.co.uk  CIS: 100102,145
Web: www.sthdown.demon.co.uk

Southdown House Software Ltd., Southdown House, Guildford Road, Westcott, Dorking, Surrey, RH4 3NR, U.K.
Getting the News
News Reader Software and Usenet Access

By Douglas Reilly
74040.607@compuserve.com

One of the things missing from the World Wide Web sites that have, for the most part, powered the growth of the Internet, is any interaction with other human beings. Web pages are essentially static, presenting the same information to you or me, or anyone else browsing the page. In addition, you cannot get that last little bit of information you might need that is not covered on the page. A more interactive option exists.

In addition to web pages, which use Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML), the Internet allows you to access “news groups” using the Network News Transfer Protocol (NNTP). News groups are accessed a little differently than web pages. Rather than a web browser like Internet Explorer or Netscape Navigator, you use a specialized news group reader. You can use the ones built into some versions of Internet Explorer and Netscape Navigator, or a specialized news group reader, like Gravity from Annawave (http://www.annawave.com, shareware, registration US$29) or Free Agent from Forté (http://www.forteinc.com, Freeware). Free Agent has a big brother, called Agent, but I have not tried it.

A news group is sort of like a bulletin board, where people post questions and others answer the questions. Since everyone posting all kinds of questions together would be chaotic, there are thousands of news groups that each specializes in one subject area. Recently, as mentioned on page 17, Btrieve got its own newsgroup, comp.databases.btrieve. Not all Internet service provider’s news servers make all news groups available. Mine does not allow me access to comp.databases.btrieve yet, but I follow most threads using www.dejanews.com.

One thing to be aware of when reading through the news groups: The information contained should be taken as if it were simply posted on a bulletin board. There is no assurance that the information is reliable, and often the folks posting might have an axe to grind. In addition, contrary to the normal etiquette, some folks “cross-post,” sending a message about, say, a new C++ compiler to dozens of news groups, including some that have no relation to C++. In addition, the groups available cover a wide spectrum of interests and tastes (yes, there is a news group alt.barney.die.die.die).

One thing that can help if you want to cruise through some news groups is an enhanced newsreader, like Anawave’s Gravity. Some of the features offered by Gravity, but not present in the news readers that currently come with the Microsoft Internet Explorer and Netscape Navigator browsers, include:

- The ability to search for messages on a particular topic.
- The ability to filter messages on the basis of topic or author.
- This feature is often called a “bozo filter.” Specify a particular author (perhaps one who drones on and on about a product produced by his or her company) or a text string (I filter on the phrase “Easy Money”) and you will not even see those messages. Without such filters, any public group can have a very high level of noise versus signal. Sometimes it would be nice to have a bozo filter in “the real world.”
- The ability to automatically signal you when a particular type of message is found. When my daughter was first sick with pancreatitis, I had Gravity read through several newsgroups and beep whenever it found something about that topic.

In my personal experience, the newsreaders included with the browsers are barely serviceable. Both Free Agent and Gravity improve on the basics and add something, but I have not found anything better than Gravity. One disadvantage of most newsreaders over the one included with Internet Explorer is the inability to connect to more than one news server per session. If you have more than one news server you connect to regularly (as I do) this can be a drawback. Happy news reading!

Smithware Launches Electronic Mailing List for Btrieve Development Alerts

NASHVILLE, Tennessee — Smithware, publisher of Btrieve Developer’s Journal magazine and Btrieve-based development tools, has launched a new on-line news distribution service called the SmithwareDevWire.

The e-mailing list is hosted at the Smithware web site in Nashville, and is open to all parties interested in receiving the latest information about Btrieve and Smithware development issues. According to Smithware President and BDJ Publisher, Scott Smith, “The SmithwareDevWire news service is the best way for us to get information out on Btrieve related development issues quickly.”

To join the SmithwareDevWire mailing list, send an e-mail addressed to listserver@smithware.com with “Subscribe SmithwareDevWire” as the message subject.

The SmithwareDevWire news service is one of many enhancements to its on-line support operations that Smithware recently has introduced. In May, the company unveiled a new searchable on-line knowledge base located at http://www.smithware.com/support/kbase.

The technology used to create SmithwareDevWire is being introduced this summer as a new product by Smithware called the Exchange ListServer Agent. More information is available at http://www.smithware.com/listservice.
When To Use Scalable SQL
When To Use Btrieve?

Richard B. Trocino
rtrocino@io.com

Pervasive Software offers two extremely flexible and useful databases. And sometimes it is not immediately apparent when you should use Btrieve and when Scalable SQL might do a better job.

The MicroKernel Database Engine (MKDE) is the foundation of Pervasive Software’s database architecture. The MKDE performs all of the low-level data manipulation including indexing, caching, transaction processing, and multi-user controls. Btrieve and Scalable SQL serve as access modules that sit on top of the MKDE.

Pervasive Software offers Btrieve and Scalable SQL as separate products. Both of these products include the MKDE. Btrieve and Scalable SQL are both available in client/server versions for middle market network applications (those with less than 1,000 users) as well as workstation versions for standalone personal computers.

Pervasive Software offers developer kits for both Btrieve and Scalable SQL, enabling programmers to embed high performance database functionality into their application programs. For example, Solomon accounting software is written at the Scalable SQL level and uses the Scalable SQL API set. Platinum, on the other hand, is written at the Btrieve level and uses the Btrieve API.

If you are a software developer, the question, “Which one do I use?” becomes more complicated. The answer is “either,” but probably not both within the same application. If you are maintaining an existing Btrieve application, you probably want to stick to Btrieve. To offer SQL access, all you have to do is ensure that you supply up-to-date DDF files and let the network administrators sort out the rest. If your application is already based on SQL, then use the Scalable SQL API set and write your program to the Scalable SQL API.

Which access method should you use if you are starting a new project? The answer, again, depends on you. What are your priorities? I have found there are three general areas of concern in the world of database computing: performance, compliance, and convenience. If your main priority is performance, then choose Btrieve. If your main concern is compliance or convenience, then choose Scalable SQL.

I don’t mean “compliance” and “convenience” in a derogatory way. Many industries, agencies, and companies have regulations covering database methodologies. For example, a government agency may mandate that all database engines must comply with standards of security and relational integrity. Another example of compliance is a company standardizing on the relational model of data with formalized rules on redundancy and normalization. In the business world, standards of compliance are usually based on SQL, the official language of relational theory. This means the ANSI standard for SQL contains syntax that address most of the compliance requirements a company or agency may have.

Sometimes you are just writing “throw away” programs to perform simple functions or fix specific problems. In this case, convenience is your primary concern, so keep it simple. Again, SQL might be your best choice. SQL syntax can be a lot less complicated than Btrieve code. Sometimes it is far easier to write a few SQL scripts, than to write and compile a Btrieve program.

Performance is a different question all together. How fast can my application deliver data on a busy network? Some people are very performance oriented, sometimes to the point of obsession. Programmers who are concerned with performance above all else are the ones that use Btrieve. They will always attempt to achieve peak performance, sometimes at any expense. Just throwing faster CPUs, more memory, and better disk drives at the problem won’t cut it. These programmers are after the theoretically optimal solution: the perfect algorithm with the fastest throughput.

I have been racing sailboats, in one capacity or another for a long time. I never cease to be amazed by the lengths to which some skippers will go to achieve optimal performance. I was at the marina the other day and a racing buddy of mine had pulled his boat out of the water, hoisted it up, and was sanding the bottom with an electric sander.

“You see these little bumps?” He said pointing to what looked like a solidified drip of paint that was smaller than the tip of my little finger. “They develop from time-to-time and you have to sand them off,” he said. There were about seven such bumps on an otherwise immaculately clean and smooth hull. “It all adds up,” he said.

Whether you are racing sailboats, competing in a triathlon, or developing a multi-user database application, if performance is your obsession, you have to exploit every advantage that you can, reduce drag whenever possible, and always push for speed. In this case, Btrieve is the weapon of choice.

Stay tuned for the next article, Sanding the Bumps and tips on how to eke-out more performance with your client/server Btrieve applications for NetWare and Windows NT Server.
aspect of the entire application. Granted, part of the ease of use was due to my familiarity with the underlying classes, but the fact was that I spent less than two hours (six percent of the project time) writing Btrieve code. How does this happen?

First, let’s look at what a Java applet needs in terms of Btrieve support. First, it needs a way to get to the Btrieve API. Then, it needs buffer handling mechanisms; along with those come the requirements for field extraction and type conversion. Since Java uses a different byte-ordering scheme than Btrieve (or any Intel-based database), there must also be support for byte re-ordering when getting data from the buffer. Furthermore, there needs to be some base support for data-type conversion. Because I’m not a masochist, I really prefer to have a helper function or class for extended operations, as well. Finally, there are the essentials such as communications optimization, server security, and compression to consider.

Fortunately, all of these features and more are built into the Smithware Java class library and the IDS. The amount of code needed to get from LAN to Internet in Java is minimal in size and effort. The Java client classes are flexible, so I chose to go with the most developer support at the expense of a slightly enlarged footprint. Because Java does not allow for multiple inheritance, a functional Btrieve class must use one of three levels of Btrieve support, each of which builds on the other.

The first and lowest level package (a Java package is a special group of related files) provides for Btrieve access, security, and data compression. This package has a footprint of about 13K, but is not particularly easy to use. It provides no buffer management and performs little optimization other than data compression.

The second package is ideal for applications that do not need extended operations. In addition to providing buffer management and per-call optimization, it also manages data type conversion for most data types. It weighs in at a sweet 9K, which is added to the 13K of the base class for a total size of about 22K. Per-call optimization means that the class will determine the correct send and receive buffer length based on the Btrieve call being made. For instance, a GetDirect call only needs to send four bytes, but may need to return 4K. Since the IDS can receive and return different buffer lengths, the GetDirect call is optimized by this package to only send a small buffer but to be prepared for a large return buffer. This package includes a helper class for handling COBOL key types; these types are too infrequent and require too much overhead (another 6K) to justify inclusion in the general buffer class.

The third and final package is optimized for extended operations. Although the footprint adds another 9K to the size of the applet, the price is well worth it when using extended operations. It maintains a cache and makes setting up extended operations call almost fun!*

Needless to say, this applet uses all three packages (excluding COBOL support) and thus requires about 32K of downloaded applet code to run. However, when compared to some of the web page wallpaper out there, 32K is not an unreasonable burden. In fact, two of the Symantec classes used for this applet come in at 17K apiece.

There are several steps to getting a Java applet or application to read data using the IDS, the first of which is subclassing the desired client class (one of the three levels of Btrieve support). The reason that the subclassing is necessary is because the security functions are abstract, meaning that they have not been defined. For an applet communicating with an unsecured site (such as the Smithware test server), these functions can remain stubs, since they will never be called. For secured sites, these functions must each return a portion of the login information. For most clients, the buffer class will be the class of choice, while apps needing extended operations will want to subclass the ExOps class. For this applet, I used both.

The first step is to declare the classes; such a declaration might be as follows (I put this in the applet class itself):

```java
private SBufferClient m_userFile = new SBufferClient(); //non-Exops client
private SExOpClient m_allUsersFile = new SExOpClient(); //Exops client
```

The next thing to do is initialize the classes. Since the classes must know record and key size in order to optimize calls effectively, I include the following code for each class before attempting to connect to the IDS:

```java
m_userFile.SetCompression(1); //Turn compression on (default is on, so this is redundant)
m_userFile.SetMaxSize(125); //max record size
m_userFile.SetKeyNumber(1); //set key number (default is 0)
```

Then, in the initialization or connect function, the applet should connect to the IDS, after which it can try to open a file on the IDS. Since every call to the IDS can throw an IOException exception, the entire process can be put in one try...catch block. I’ll only reproduce one of my connect calls, since they are all identical (I put this in applet.init()):

```java
try
{
    //This will call our security functions if IDS LogonRequired = TRUE
    m_userFile.SWConnectAndLogon("207.65.21.114");
    //write the file name to the key buffer and the
    //owner name to the data buffer
    m_userFile.WriteString(new String("<filename>"), 0, 0, SBufferClient.USE_KB);
    m_userFile.WriteZString(new String("<owner name>"), 0, 9, SBufferClient.USE_DB);
    //open the remote data file
    m_stat = m_userFile.BTRV(0);
   ...
do some error handling - if this is hit, the session is completely busted
}
catch (IOException e)
{
    ...
do some error handling
}
```

So, in eight lines of actual code (nine, if you count the declaration line), I’ve managed to connect to the IDS and open an owner-name protected file. The WriteString() and WriteZString() functions place the data in the appropriate buffer (the middle parameters for each call are offset and length, respectively — a length of zero in WriteString() tells the function to use the length of the string itself). In my opinion, the coolest thing about this sequence is the BTRV call: one parameter. Count them. Heck, count them again. You can count them 15 times and still have five toes left to work with! This is much nicer than the error-prone six parameters, and the buffer class takes care of buffer optimization for you.

From then on, making a non-extended operations call is a simple try...catch block around the BTRV function, such as:

```java
try
{
    m_stat = m_userFile.BTRV(12); //GetFirst
    ...
do some error handling
}
catch (IOException e)
{
    ...
do some error handling
}
```

Getting data out is simply a matter of calling the various read functions along with the offset, the length (if necessary), and the location (data buffer or key buffer) of the field. For example, the code that loads the first dialog page with user information is as follows:

```java
```
Pretty tricky, eh? Getting the data out of the buffer is very similar to the calls seen previously, with the exception of the ConvertOffset() function. ConvertOffset() takes a record number and a record offset and converts it to an absolute offset of the data buffer based on the most recent ExOp call. The code to extract the first and last names from the tenth record, therefore, would look like:

There’s only one other aspect to cover: selecting records based on field criteria. This is very similar to the field selection process, with the exception that the selection criteria are joined by logical conjunctions.

Again, this assumes that file positioning has been established. Furthermore, each of these code snippets is run inside a synchronized{} block to prevent thread corruption. While this may look like several lines of code, I’ll point out that 1) programming the same task in C would take at least 10 times as much code 2) programming the same task in C would require hours of debugging 3) most of this code is cut-and-paste stuff and 4) that’s what you get paid to do. Furthermore, any other existing technique of accessing this data through the web will be less efficient, more error-prone, and much more painful (can you say server-side scripting, ActiveX, CGI, and ODBC? I knew you could, but why would you want to?)!

Although the web is often not the medium of choice for real work, for a publicly available, low-investment applet such as this, the web is ideal. The problems with the web as far a program platform are reduced drastically by the use of Java, and the IDS class libraries are the best way for Java to access Btrieve.

(The ReadLogical field takes as the third parameter the “True” value stored in the field.)

Getting an extended operations call correct is only slightly more difficult, as it requires the use of a couple of helper classes: one for the selected fields and one for the filter conditions. Since the applet uses both of these (in different classes), I’ll demonstrate each.

First, the first thread oﬀshoot to run performs a download of all the users on ﬁle; this was done to retain normalization without the expense of redundant open calls over a low-bandwidth connection. Because it’s in a separate thread, with any luck this thread should complete as the user is logging on. I only wanted to get one cache, so I selected only those ﬁelds I needed for this purpose: user number, ﬁrst name, and last name. Each ﬁeld requires the instantiation of a small class (this would be a structure in C, but hey — this is Java) which will be passed to the ExOps class. The code to create the ﬁeld classes is as follows:

The SetVariable() call tells the ExOps class whether or not the extracted record contains any variable length ﬁelds. If there are no variable length ﬁelds, getting data from the ExOp buffer will proceed signiﬁcantly faster than if there are. ClearAll() clears both ﬁeld and selection criteria from the ExOps buffer. Finally, we’re ready to make the call, which must be included in an try...catch block:

(The discussion group applet described here can be accessed online at http://www.smithware.com/discussion/idsdiscussion.html, And see http://www.smithware.com/ids for complete information on the Smithware IDS and 1*Net developers kit.

This is pretty much cut-and-paste work. The next thing is to initialize the ExOps class and identify these fields as the fields to be retrieved. By this time, I’ve already established my starting position with a GetFirst, as demonstrated previously.

The SetVariable() call tells the ExOps class whether or not the extracted record contains any variable length fields. If there are no variable length fields, getting data from the ExOp buffer will proceed significantly faster than if there are. ClearAll() clears both field and selection criteria from the ExOps buffer. Finally, we’re ready to make the call, which must be included in an try...catch block:

Pretty tricky, eh? Getting the data out of the buffer is very similar to the calls seen previously, with the exception of the ConvertOffset() function. ConvertOffset() takes a record number and a record offset and converts it to an absolute offset of the data buffer based on the most recent ExOp call. The code to extract the first and last names from the tenth record, therefore, would look like:

There’s only one other aspect to cover: selecting records based on field criteria. This is very similar to the field selection process, with the exception that the selection criteria are joined by logical conjunctions.

Again, this assumes that file positioning has been established. Furthermore, each of these code snippets is run inside a synchronized{} block to prevent thread corruption. While this may look like several lines of code, I’ll point out that 1) programming the same task in C would take at least 10 times as much code 2) programming the same task in C would require hours of debugging 3) most of this code is cut-and-paste stuff and 4) that’s what you get paid to do. Furthermore, any other existing technique of accessing this data through the web will be less efficient, more error-prone, and much more painful (can you say server-side scripting, ActiveX, CGI, and ODBC? I knew you could, but why would you want to?)!

Although the web is often not the medium of choice for real work, for a publicly available, low-investment applet such as this, the web is ideal. The problems with the web as far a program platform are reduced drastically by the use of Java, and the IDS class libraries are the best way for Java to access Btrieve.

(The discussion group applet described here can be accessed online at http://www.smithware.com/discussion/idsdiscussion.html, And see http://www.smithware.com/ids for complete information on the Smithware IDS and 1*Net developers kit.

This is pretty much cut-and-paste work. The next thing is to initialize the ExOps class and identify these fields as the fields to be retrieved. By this time, I’ve already established my starting position with a GetFirst, as demonstrated previously.

The SetVariable() call tells the ExOps class whether or not the extracted record contains any variable length fields. If there are no variable length fields, getting data from the ExOp buffer will proceed significantly faster than if there are. ClearAll() clears both field and selection criteria from the ExOps buffer. Finally, we’re ready to make the call, which must be included in an try...catch block:

Pretty tricky, eh? Getting the data out of the buffer is very similar to the calls seen previously, with the exception of the ConvertOffset() function. ConvertOffset() takes a record number and a record offset and converts it to an absolute offset of the data buffer based on the most recent ExOp call. The code to extract the first and last names from the tenth record, therefore, would look like:

There’s only one other aspect to cover: selecting records based on field criteria. This is very similar to the field selection process, with the exception that the selection criteria are joined by logical conjunctions.

Again, this assumes that file positioning has been established. Furthermore, each of these code snippets is run inside a synchronized{} block to prevent thread corruption. While this may look like several lines of code, I’ll point out that 1) programming the same task in C would take at least 10 times as much code 2) programming the same task in C would require hours of debugging 3) most of this code is cut-and-paste stuff and 4) that’s what you get paid to do. Furthermore, any other existing technique of accessing this data through the web will be less efficient, more error-prone, and much more painful (can you say server-side scripting, ActiveX, CGI, and ODBC? I knew you could, but why would you want to?)!

Although the web is often not the medium of choice for real work, for a publicly available, low-investment applet such as this, the web is ideal. The problems with the web as far a program platform are reduced drastically by the use of Java, and the IDS class libraries are the best way for Java to access Btrieve.

(The discussion group applet described here can be accessed online at http://www.smithware.com/discussion/idsdiscussion.html, And see http://www.smithware.com/ids for complete information on the Smithware IDS and 1*Net developers kit.

This is pretty much cut-and-paste work. The next thing is to initialize the ExOps class and identify these fields as the fields to be retrieved. By this time, I’ve already established my starting position with a GetFirst, as demonstrated previously.

The SetVariable() call tells the ExOps class whether or not the extracted record contains any variable length fields. If there are no variable length fields, getting data from the ExOp buffer will proceed significantly faster than if there are. ClearAll() clears both field and selection criteria from the ExOps buffer. Finally, we’re ready to make the call, which must be included in an try...catch block:

Pretty tricky, eh? Getting the data out of the buffer is very similar to the calls seen previously, with the exception of the ConvertOffset() function. ConvertOffset() takes a record number and a record offset and converts it to an absolute offset of the data buffer based on the most recent ExOp call. The code to extract the first and last names from the tenth record, therefore, would look like:

There’s only one other aspect to cover: selecting records based on field criteria. This is very similar to the field selection process, with the exception that the selection criteria are joined by logical conjunctions.

Again, this assumes that file positioning has been established. Furthermore, each of these code snippets is run inside a synchronized{} block to prevent thread corruption. While this may look like several lines of code, I’ll point out that 1) programming the same task in C would take at least 10 times as much code 2) programming the same task in C would require hours of debugging 3) most of this code is cut-and-paste stuff and 4) that’s what you get paid to do. Furthermore, any other existing technique of accessing this data through the web will be less efficient, more error-prone, and much more painful (can you say server-side scripting, ActiveX, CGI, and ODBC? I knew you could, but why would you want to?)!

Although the web is often not the medium of choice for real work, for a publicly available, low-investment applet such as this, the web is ideal. The problems with the web as far a program platform are reduced drastically by the use of Java, and the IDS class libraries are the best way for Java to access Btrieve.
Making ODBC Work

Richard B. Trocino
rtrocino@io.com

The ODBC driver represents a strategic part of Pervasive Software’s product line. ODBC is a SQL-based database driver that provides a common interface to data. Most major database engines include an ODBC driver. This allows application programs to connect to a variety of dissimilar databases using one standard interface.

The MicroKernel Database Engine (MKDE) provides the foundation for Pervasive’s new database architecture and performs all of the physical data file manipulation. ODBC is one of the three primary access models for accessing the MKDE—the other two are Btrieve for navigational access and Scalable SQL for relational access. The ODBC access module is really a crossbreed of the relational and navigational access models, capable of using software components from the other two models.

ODBC is primarily useful for integrating MKDE files (those generated by Btrieve and Scalable SQL) with third-party software such as Microsoft Access, Microsoft Word, and Microsoft Excel. The true value of ODBC is providing relatively simple access to mission-critical data using familiar tools. Generally, ODBC is not considered the best choice for the mission-critical applications themselves — this is better left to Btrieve or Scalable SQL.

In my experience, the ODBC driver version 2.0 works quite well and is a vast improvement over earlier versions. This problem is that it doesn’t always install in a manner suitable to your particular environment. And the software components (specifically the requesters and related DLLs) aren’t always put in the most convenient place. In many cases, the initial installation of ODBC is unsuccessful, leading to a hasty assessment that “the product doesn’t work.” However, installation problems are simply solved if you understand a few important points.

There are two basic steps to installing ODBC:

1) installing and configuring the software components (requesters, drivers, and utilities), and

2) defining a data source to ODBC by pointing to an existing data dictionary.

Once the software components are installed properly, the second step is to use the ODBC Administrator from the Windows Control Panel to define a data source. This step involves making sure your data dictionary is properly in place. A data dictionary is a collection of files referred to as data dictionary files (DDFs). ODBC views your data dictionary as a data source that is defined to the ODBC driver. When you run an ODBC application, you simply select from a list of data sources and link to a database. This article assumes that you know how to connect to an ODBC data source through your ODBC-compliant application. I used Microsoft Access version 7 and created a link table when writing this article.

Data Dictionary Files

When debugging an ODBC application, divide your problem into the two above steps. Is your problem with the ODBC installation or is it with your data dictionary? It is important not to confuse the two. I recommend using the DEMODATA dictionary supplied with the ODBC driver. DEMODATA is a fresh set of DDFs and accompanying database files. (The minimum files for a valid data dictionary are FILE.DDF, FIELD.DDF, and INDEX.DDF) Try to make ODBC run with the dictionary supplied by Pervasive. Once you can do this, then attempt to connect to your own DDFs.

This article assumes that your data dictionary is properly defined and can be recognized by the ODBC driver. Most major software packages include DDFs with their product. If your software does not include DDFs, they must be obtained from the software developer or you must created them yourself. DDFs are created with Scalable SQL or a data dictionary utility such as Smithware DDF Builder.

A description of how to set up data dictionaries can be quite lengthy, so I will address it in a subsequent article.

Architecture Options

This article considers three basic architecture options for ODBC. These options each have their own characteristics and are listed in order of their relative performance.

Three Architectural Options

ODBC with client/server Scalable SQL 4
ODBC with client/server Btrieve
ODBC with workstation Btrieve

By far, the highest performer of the three architecture options is ODBC with client/server Scalable SQL 4. SQL parsing and generation of the access plan is performed at the file server by the Scalable SQL engine. The access plan is the algorithm that Scalable SQL uses to perform a given query or table update. In addition, data is passed to the workstation in multiple-record blocks. These factors dramatically reduce network traffic and can vastly improve performance on a busy network.

The second option, ODBC with client/server Btrieve uses the Btrieve requester. SQL parsing is performed at the workstation. Remember that ODBC uses SQL, so parsing has to be done somewhere. The SQL statement is reduced to a series of Btrieve operations that are then processed by the server-based MKDE.

The third option, ODBC with workstation Btrieve also performs SQL parsing at the workstation, but then calls the workstation version of the MKDE. This configuration is primarily intended for standalone PCs and small networks with fewer than five users. The ODBC workstation Btrieve configuration will operate on server-based networks that are not running client/server Btrieve. However, on a busy network, performance may leave a little to be desired.

If you can use the client/server Scalable SQL configuration, then by all means do so. In many cases, the need for solid ODBC performance justifies the additional cost of Scalable SQL. If you are supporting an existing Btrieve application and don’t have Scalable SQL, then you must use one of the Btrieve configuration options, preferably client/server. My recommendation is, on a network, use client/server whenever possible and really take advantage of the technology.

32- and 16-bit Software Components

In any of the three architecture options described above, you can use either 16-bit or 32-bit software components on the workstations.
Windows 95 and Windows NT Workstation are 32-bit operating systems capable of using the 32-bit software components and drivers. Windows 3.x is a 16-bit operating system, so it must use the 16-bit ODBC driver. 32-bit applications like Microsoft Access usually use the 32-bit ODBC driver, while 16-bit applications use the 16-bit ODBC driver.

Thunking is a technique that enables 16-bit applications to use 32-bit software components. However, I would avoid the thunking option if you can. An application program is better off running with pure 32-bit or pure 16-bit software components and drivers.

Configuration Options

Since there are three architectural options and two software component options, there are six possible configuration options available for an application using ODBC. These are 16- and 32-bit configurations for client/server Scalable SQL, client/server Btrieve, and workstation Btrieve.

The following section presents a step-by-step approach to configuring ODBC in your particular environment. Each of the three architectural options is presented with instructions on configuring both 32-bit and 16-bit applications.

If you are having trouble making ODBC work, identify which of the above configuration options fits your scenario and then skip to the section below that describes your particular situation. For example, if you are a Solomon user using Scalable SQL 3.x as well as client/server Btrieve 6.15, go to the second architecture option, ODBC with client/server Btrieve. If you cannot fit yourself into one of these six categories, you may be trying to do something that the ODBC product was not intended to do.

Configuration Settings

Any given ODBC driver will look for the configuration settings in one particular place. For 32-bit applications, the configuration settings are stored in the Windows Registry and modified with the Pervasive Software 32-bit Setup program installed in step two. If you are a Windows purest, you can also use the registry editor utility (REGEDIT.EXE) to modify the registry directly.

For 16-bit applications the configuration settings are stored in the BTI.INI file in the Windows main directory and modified by either the 16-bit Setup program or a text editor. Remember, Windows 3.x is a 16-bit operating system, so it uses the 16-bit ODBC driver. And Windows 95 and Windows NT are 32-bit operating systems, so they can use either the 32-bit or 16-bit driver. When configuring ODBC you must know whether your application is a 32-bit or a 16-bit application, then choose the appropriate ODBC configuration.

Architectural Option One: ODBC with Client/server Scalable SQL 4.0

This option offers the best performance of the three architectural options. In this configuration, the SQL requesters are used and all SQL processing is handled at the file server by Scalable SQL. As a result, network traffic is minimized — only SQL statements and requested data are transmitted over the network.

Step 1: Install SQL on the server

Install Scalable SQL on your server from the Setup Routine on the distribution CD. Before using ODBC, I recommend demonstrating a proper installation of Scalable SQL by logging into the database supplied with Scalable SQL. By default, this database is installed into the $SQL40$DEMODATA directory. Step five describes this process.

Step 2: Install the Workstation Components

Install the workstation components by running the Setup program located in the $SQL40$INSTALL directory. You may have installed Scalable SQL in a directory other than $SQL40$; if so, the INSTALL directory will be in that directory. Install the 32-bit requesters and utilities and the 32-bit ODBC driver.

Step 3: Set the configuration options

Set the proper configuration options on the workstation. Make sure the following settings are made using the following tables.

### Setup Parameters for 32-bit ODBC

<table>
<thead>
<tr>
<th>Setting in the 32-bit SQL40 Setup program</th>
<th>Registry Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL-Requester/Access Control/Local Usage='OFF'</td>
<td>HKEY_LOCAL_MACHINE/Software/Pervasive Software/Scalable SQL Requester/Version 4.00/Settings/Local='no' Requester='yes'</td>
</tr>
<tr>
<td>Remote Usage='ON'</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting in the 16-bit SQL40 Setup program</th>
<th>BTI.INI setting in Windows main directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL-Requester/Access Control/Local Usage='OFF'</td>
<td>[Scalable SQL] local=no requester=yes</td>
</tr>
<tr>
<td>Remote Usage='ON'</td>
<td></td>
</tr>
</tbody>
</table>

### Setup Parameters for 16-bit ODBC

Step 4: Set a path to your requesters

Make sure that your system can find the Scalable SQL requester and accompanying files. The ODBC installation will place the requesters and other software components on your workstation in the C:\$SQL40$BIN directory by default. I find, however, that it is better to run your requesters and other software components from the server. This is accomplished by setting the PATH or searched mapped drive on each workstation to the $SQL40$BIN directory on your server. If you do this with all your workstations, you can ensure that everybody is running the correct version of the requesters. When a workstation is operating in a purely client/server environment, I actually remove all of the requesters and related software components from the workstation. Specifically, these files are the following:

- **32-bit workstation software components used with client/server Scalable SQL 4.0**
  - WSQSQL32.DLL
  - W32BTICM.DLL

- **16-bit workstation software components used with client/server Scalable SQL 4.0**
  - WXQLCALL.DLL
  - WBTICOMM.DLL

Step 5: Make sure you can access the DEMODATA database

At this point, it is important to demonstrate a proper implementation of your software components. I recommend using SQL Scope to log into the database called DEMODATA supplied with Scalable SQL 4.0. Doing so ensures that everything is working at the client/server SQL level. Remember, at this time, SQL Scope is only available in a 16-bit version so you have to use the 16-bit setup utility to configure SQL Scope.

See ODBC, page 26
I would like to know if it is possible to run a Btrieve database on a DOS machine and then access the information from a Windows ‘95 or NT workstation?

B.K.
South Africa

The Colonel Responds:

Sweet petunias! What dusty old barn did you find a DOS machine in? I assume we’re talking good old-fashioned DOS, as in MS-DOS version six or previous, as opposed to one of those nifty little “DOS sessions” running underneath a real operating system? I’m almost certain it’s possible, but it’d probably be about as easy to rig a Sopwith Camel for in-flight refueling behind a KC-135.

Your first challenge would be to try wiring the two machines together (assuming, of course, that you’re not planning to carry five-and-a-quarter-inch floppies back and forth all day). A pair of jumper cables or a suitable length of 14-gauge Romex strung between the NIC on the Windows NT workstation bus and those two little brass posts on the back of the DOS-based IBM AT where the Morse Code key normally attaches should do the trick. Of course you’ll need to put what’s called in technical terms a “little black box,” or LBB, in the line to handle protocol translation and handshaking. But you can either requisition one of those up at your local LBB store or build one yourself using a scrapped alternator from a ‘57 Chevy and a few common vacuum tubes. Fortunately a friend of mine sent me the schematic for the one he used to connect HP-9845 desktop calculators and Compaq 286 systems in the mid 80’s. I’d reprint it here but the dog got ahold of it and now the typesetter won’t touch it. The ink ran some in places but you should be able to figure it out pretty easily. Just check your fire insurance policy for exclusions first, and be careful to get the impedance right between the secondary coil and the anode of V2. I can’t really tell whether that first number used to be a three or an eight, and it may have been a resistor and not a really number at all, but it certainly does appear to be important.

Anyway once you’ve established a good electrical connection, all you have to do is write a little peer-to-peer networking code, which shouldn’t take more than a few thousand hours at the outside. Once that’s done you’re ready. The file formats used by the DOS and Windows NT versions of the Btrieve MicroKernel Database Engine are interchangeable, so you shouldn’t have any problem at all sharing the data between systems. And it won’t cost you a dime.

There’s one other option that I almost hesitate to mention, since it lacks both the elements of intellectual challenge and mortal peril. But I suppose you could go out and buy a network operating system. I’m not recommending one over another, but Novell NetWare has established a pretty good track record for accommodating mixed hardware and workstation OS arrangements, and you don’t have to have a 400Mhz dual Pentium with 8Gig in RAM on board to run it. It sounds pretty low-tech to mention it these days, when everybody’s carrying on about I*Net this and Web that, but you can still get most of the parts off the shelf to set yourself up a little LAN. If you put your Btrieve files on the server, all the machines on the LAN will be able to access them, regardless of what operating system the individual workstations are running, as long as they can map the network drive. Even better, you can load a Btrieve server engine on the file server, and then use the appropriate Btrieve requesters at each workstation. It is a little more expensive and a lot less interesting than the do-it-yourself approach. But it does, by some salutary coincidence, happen to be a fairly practical and efficient way to run a DOS application on one workstation and a Windows application on another and share the same data. It’s almost as if they designed it that way.

*Ed. note: For a more practical treatment of the issues involved in running DOS applications on Windows NT clients, see the FYI article Support of DOS Applications on NT Clients, in the Pervasive Software Technical Corner section on page 13 of this issue.*

Col Micro, Sir:

I was wondering if you could help me with this question. Is Btrieve year 2000 compliant? Many thanks for your help.

A.H.
Coventry, U.K.

The Colonel Responds:

Just what part of “yes” didn’t you understand last time I fielded this preposterous question? Maybe I didn’t make myself perfectly clear, so I’ll go over it again, very slowly. Is Btrieve year 2000 compliant?

Absolutely, positively, beyond a shadow of a doubt, yes.
Which is the right answer to the wrong question. The right question is “Will my application that uses Btrieve break in three years?” And I can’t answer that one for you, except to say that just because you have to ask I’d be really surprised if it didn’t. When it does, though, you won’t be able to blame Btrieve. A couple other factors are involved answering the real question. First, applications can do a lot of things with data that have nothing to do with how the data is stored or retrieved, and if the application code deals with dates without regard to the century, chances are that code is destined to fail.

Soon. Second, Btrieve does not dictate the way in which an application stores its data, and if the application stores dates without the century component, and then asks Btrieve to sort it, well, which do you think is greater, zero or 99? Would you really want the database management layer of your application deciding that zero is greater than 99 in certain cases? If so, what might those cases be? I’m sure the development team at Pervasive Software would be open to suggestions for improvement.

For over a decade the Btrieve Developer Kits have hinted at how you ought to store a date value, and have provided a key type for dates that’s pretty much guaranteed compliant for at least the next 30 thousand years. Year as two-byte integer. Range: 64 thousand years plus tax and gratitude. I admit I have no earthly idea whether dates BCE collate correctly or if the thing’s treated as unsigned, nor do I particularly give a rat’s posterior. Month as one-byte integer. Range: either 0 to 255 or -127 to +127, not that it makes any difference since we’ve only identified twelve of them so far. Day as one-byte integer. Ditto. If you use the date key type and you store the right date, you can count on Btrieve to sort the date right. Yes, there’s still what you might call an inherent bug, but at least we can safely put off worrying about it again until mid-summer 32764, by which time I really do hope to be spending my afternoons fishing.

Btrieve is Y2K compliant, but you may not assume that just because a given application uses Btrieve, that the application is therefore Y2K compliant. Btrieve wastes very little time attempting to protect a programmer from his or her own stupidity. It will catch attempts to insert duplicate values for unique keys, or to modify non-modifiable key values, and it will detect and thwart efforts by two different processes to modify the same record at the same time.

Beyond such basic constraints of good record management, Btrieve will leave you a great deal of latitude, in which you may take aim at your left foot and fire at will. Garbage in will produce just what you’d expect. But you certainly can, and at this late date I would go so far as to say that you probably should, write Y2K compliant applications using Btrieve.

Save $30

Start Your Subscription or Renew Today and Get Special Subscriber Discounts!

Take advantage of special subscriber discounts on Btrieve Complete and The Illustrated Guide to NetWare Btrieve. These two great reference books should be part of every Btrieve and Scalable SQL developer’s library. They contain a wealth of up-to-date information, including tips for understanding, configuring, and programming the Btrieve record manager. For a limited time, you can save up to $30 when you begin or renew your subscription to Btrieve Developer’s Journal. Both books retail for $39.95, but with your paid subscription to BDJ, get either book for $24.95, a $15 savings off the cover price. Buy both, and save $30!

Offer good only in conjunction with new subscriptions and renewals. Book will be shipped when payment is received. Limit one copy of each book per subscriber.
Use the following settings:

<table>
<thead>
<tr>
<th>Setup Parameters for 16-bit SQL Scope</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting in the 16-bit SQL40 Setup</td>
<td>BTI.INI setting in Windows main program directory</td>
</tr>
<tr>
<td>Local Usage='OFF'</td>
<td>[Scalable SQL]</td>
</tr>
<tr>
<td>Remote Usage='ON'</td>
<td>requester=yes</td>
</tr>
<tr>
<td>Thunking='OFF'</td>
<td>thunk=no</td>
</tr>
<tr>
<td>Local Usage='OFF'</td>
<td>Local Usage=off</td>
</tr>
<tr>
<td>Remote Usage='ON'</td>
<td>Remote Usage='on'</td>
</tr>
<tr>
<td>Use Thunk='OFF'</td>
<td>thunk=no</td>
</tr>
</tbody>
</table>

Don’t even attempt to make ODBC work if you can’t log into DEMODATA or any other database with SQL Scope. If you receive a 2103 error from SQL Scope, this means that Scalable SQL itself is improperly configured on the server, or the workstation components are malfunctioning. Ask your tech support person for the White Paper titled “2103 Errors with Scalable SQL” for tips on how to correct this problem. Chances are your configuration settings aren’t right, your protocol parameters in Control Panel/Network are incorrect, the workstation can’t find the requesters, or the workstation is using the wrong requesters.

**Step 6 Define DEMODATA as an ODBC datasource**

Now use the ODBC Administrator from the control panel to define the DEMODATA DDFs as a data source. At this point ODBC will attempt to connect to your database. If you can connect here, you will probably be able to connect inside your ODBC application.

Once you can successfully log into DEMODATA through ODBC, you can claim victory for the moment.

**Step 7 Log into your own database**

Now that you have proven to yourself that ODBC does in fact work. Try logging in to your own database. If you have problems doing this, at least you know that the problems are with your DDFs and not your implementation of ODBC. Contact your software developer or wait until the next issue of *BDJ* for a detailed discussion on **Making your DDFs work with ODBC**.

**Architectural Option Two: ODBC with Client/server Btrieve SQL**

This option uses the Btrieve requesters and client/server Btrieve. All MKDE processing is performed at the file server, however, SQL parsing is performed at the workstation. The workstation then makes calls to the Btrieve client/server engine. As a result, this option generates more network traffic than the SQL requester architectural option described above.

**Step 1: Make Sure client/server Btrieve is properly loaded on your server**

In many cases, when you install the ODBC Driver for the first time, client/server Btrieve v6.15 or higher is already loaded on the server. If client/server Btrieve is not installed or if you are using a version lower than 6.15, you have to install client/server Btrieve v6.15 or higher and the accompanying MKDE at the server.

The installation procedure for client/server Btrieve places the 16-bit and 32-bit workstation components in separate directories. On a Windows NT Server, the 16-bit software components are placed in the `\BTI\WINNT\WIN\BIN` directory and 32-bit software components are placed in the `\BTI\WINNT\WIN32\BIN` directory. For NetWare, these directories are under `\BTI\NETWARE\`. You may want to consolidate your requesters into one directory and then have your workstations set a path to this directory containing both the 16- and 32-bit workstation components.

See **ODBC**, page 30
Btrieve and Scalable SQL 4.0

By Doug Reilly
74040.607@compuserve.com

After a very long wait, Scalable SQL version 4.0 is finally available, carrying with it the promise of a new Btrieve version that will, among other things, break the four-gig barrier on file sizes. Unfortunately, that new Btrieve interface to the MKDE that powers Scalable SQL 4.0 (SSQL4) is not ready yet. But don’t worry, you can install the SSQL4 and still access files using the MKDE that comes with SSQL4. Confused yet? Read on.

Btrieve 6.3? Did I Miss the Memo on This?

For over a year, I have been anticipating arrival of Btrieve version 7. There were many features that interested me, one of the most significant being the ability to break the 4-gig file size limit. When the SSQL4 Developers kit arrived, I thought, “I can’t wait to see what Btrieve 7 has to offer.”

Unfortunately, Btrieve 7, as a standalone product, is not yet ready, and neither is the client based SSQL4, though you can use SSQL4 on an NT machine as a very expensive client based system. Install SSQL4 on a system with a Btrieve 6.15 engine running, and a very interesting thing happens. When you next check version numbers, the Btrieve Server engine reports 6.30. How did they do that?

Btrieve 6.3 is a Shell of its Former Self

Before you install SSQL4 on a NetWare server, take a look at the size of the BRRIEVE.NLM program on the server. After the installation, look at it again, and you will see a much smaller file. This is because in this version, the only task that the Btrieve server engine is responsible for is interpreting API calls and redirecting those requests to the MKDE. There are several theoretical advantages to such an approach.

1. New interfaces to data can be added without interfering with existing data access methods. For example, neither the Btrieve API nor the SSQL API is ideal for handling Object Oriented database structure, nor for that matter even Binary Large Objects (BLOBS). Pervasive (or perhaps even a properly connected third party) could create additional data access modules that will be easier to use for these tasks.

2. All data access methods can share the same “system” services. In theory, a data access method that has no inking of Relational Integrity could be made to work within RI constraints, if those constraints were implemented at the MKDE level.

3. Upgrades can be modular, and things like user counts can be managed centrally. It is possible that upgrades that can enhance performance can be made to the MKDE level only, without interfering with the Btrieve or SSQL access method modules.

While some of these theoretical advantages have not yet been realized, there is hope that future releases will address these issues.

SSQL4 and Btrieve 6.3 Working Together

One of the options that SSQL4 offers is the ability to rebuild existing files to take advantage of the new file format. The new format was required to add features such as lifting the size restriction on files. Not surprisingly, once you rebuild the files into the new format, you can no longer access them with the 6.15 MKDE. Of course, SSQL4 will read files in previous Btrieve formats. Attempting to open a version 7 MKDE file with a Btrieve version 6.15 MKDE engine will return a status 30.

As with previous versions of SSQL/Btrieve, you can mix and match access methods on a single set of files. For instance, if you have an application written to use Btrieve as the access method, you can use SSQL to access the data from a report writer.

Are you assured of having a correctly operating system when you upgrade from a previous version of a Pervasive product to this new version of SSQL4? Not really. As discussed in last issue’s article, which I titled “DLL Madness,” the configuration management issues for not just Btrieve, but really any Windows based system can be overwhelming.

Take my personal system. I am connected to a single NT server. While I store some files on the server, the majority of what I work on is stored on the local workstation. One of the projects I am working on for a client is a Btrieve-based, 16-bit Delphi application, which uses the 16-bit Btrieve for Windows Client Engine. The customer’s copy of workstation Btrieve is installed in the application directory, in order to ensure that the application does not overwrite any other Btrieve-based application’s copy of the Btrieve DLL’s. The obvious flaw in that logic became apparent when I installed the SSQL4 32-bit client on my workstation. Upon running the program, virtually any operation that called Btrieve generated a GPF and no Btrieve reads or writes (or opens for that matter) could occur.

What I had, of course, forgotten was that both the old 16-bit engine and the new engine’s 16-bit interface use BTI.INI, an initialization file that is stored in the WINDOWS directory. Listing 1 and 2 contain BTI.615 and BTI.SQL, respectively, the BTI.INI file before the install, and the BTI.INI file after the SSQL4 client was installed. I am unsure exactly what caused

Listing 1: BTI.INI before the installation SSQL4.
Installing SSQL4

Installing SSQL4 for NT was an experience in itself. Many of you (and you know who you are) foolishly think that a graphical installation program will ensure that you can just skim the “readme” file and install the software. This is not the case with the install of the NT server version of Scalable SQL 4. Several steps need to happen, steps that one might expect the installation software would handle. None of these require rocket scientists, but they can detract from the first impressions of the product.

For instance, you need to go into the Control Panel in the Services applet and start the server after the install. If you need to go into the setup and change Target Engine settings in two places to “Server.” Another “goftcha” awaits the unsuspecting SSQL installer. Install the software. Then, suppose that you do not read the readme file, and decide to reinstall the software. If you uninstall and then delete the directories (something people commonly do to eradicate the ghosts of installs past) and then try and reinstall, you will find that the license diskette you used in the first install will no longer work, even to install on the exact same hardware. Had you read further in the readme file, you would have found the admonition to make safe copies of four files that control user counts. The files are relatively small, and perhaps a better solution would be to write the files back onto the license diskette after it is used in the first install. Licensing is an issue that causes many end users grief, and the current scheme does nothing to mitigate this. I did not have this problem only because I went to a Pervasive “Intro to SSQL4” class where a Pervasive Solution Provider had done exactly what I describe above.

Well, OK, the Install is Awkward. But How Well Does it work?

Once the software had been installed on the server and installed on the workstation, and the problems with the BTI.INI file had been, at least temporarily, resolved, I took a test drive. Using some files provided by a customer, I tried to attach tables using the ODBC 2.0 interface from Delphi 2. Two of three files did attach, but the third had a problem that previous versions of the ODBC interface seemed not to mind. The problem seems to have been that an index that appeared in the DDF files as a combination of two fields was created in the file as a single segment. This was a common practice before the advent of MS Access 1.0, which had a terrible problem attaching to such tables, but I had not seen this create a problem in some time. Scalable SQL 4 is very particular about the relationship between its data files and dictionaries, and about the structure of the table definitions and dictionary files themselves.

Once the table had been rebuilt, I set about creating a form with a grid to display the data. The normal ODBC start-up delays that you experience when you first open up a data source were longer than I normally experience on other data sources, and certain things, like reading Table Names to display in a properties box took an inordinate amount of time. On several occasions, when I went to display a list of field names for the first time I simply used that opportunity to take a break from my work. Really, I am not sure if this is a problem with the ODBC interface or the server, since I did not try any Delphi 2 work directly using the SSQL API. Given the fact that the 16-bit SQL Scope Interactive SQL utility (the only version I could find) found files and fields very quickly, and took only a few seconds to make the first access to data, I suspect the ODBC driver as the likely culprit. I will admit that the server being used was not a powerhouse (P90 with 40 meg of RAM, NT 4.0, SP 2), but the delays seen in ODBC access did seem excessive.

Conclusion

I like the idea of Server-based Stored Procedures and Triggers that this version of SSQL offers, and so far the SQL dialect that SSQL4 offers seems reasonable. Inscribe is intriguing, and I hope to find a project soon that will require use of SSQL to test out those features. I have hope for the future of this product, though I would proceed with caution before installing the NT Server version on a production server. And of course, please read that readme file!
By Jon Grieve
jgrieve@southdown.co.uk

The MicroKernel

With the release of Btrieve v6.15 came the introduction of the MicroKernel Database Engine (MKDE). This allowed a separation of the access method and the engine, meaning a common engine could exist to handle the physical access of the MicroKernel data files while the different interfaces, or models, could exist above that engine. This is illustrated in the now very familiar Figure 1.

![Figure 1 - MicroKernel Access Models](image)

In fact, this separation wasn’t quite as distinct as shown; the MKDE and Btrieve were wrapped up in a single executable file (i.e. BTRIEVE. NLM on the NetWare platform).

The Problem

Having the MKDE and Btrieve so tightly integrated never caused a problem until the introduction of Scalable SQL v3.01. Scalable SQL also requires access to the MicroKernel data files and logically needs to use the MKDE for access. Referring back to Figure 1, we can clearly see the solution; Scalable SQL is simply another interface sitting above the MKDE, right? Wrong. As we’ve already discussed, the separation isn’t that simple, and the layers aren’t so clearly defined. Figure 2 shows a more accurate diagram of the interaction.

![Figure 2 - Btrieve v6.15 and Scalable SQL v3.01 in relation to the MKDE v6](image)

It’s also interesting at this point to note the position of the arrow showing the interaction between Scalable SQL and Btrieve and the MKDE in Figure 2. It’s not 100 percent clear (at least, not to me) where the interaction takes place; Does Scalable SQL interface directly with the MKDE or does it interface with Btrieve? My guess is both.

So, with the MKDE within the Btrieve executable, it was necessary to supply Btrieve components when a customer purchased Scalable SQL v3.01. Does that mean you get Btrieve for free when you buy Scalable SQL? Not quite. Although the Scalable SQL v3.01 Server Engine does include the Btrieve v6.15 Server Engine it doesn’t include the necessary requesters. Basically, you get Btrieve running on your server but you can’t access it from any applications on your clients (i.e. workstations).

The Solution

With the introduction of Scalable SQL v4.0, all the relationships are now a great deal clearer. This is because the MKDE is no longer an integral part of Btrieve. Scalable SQL, Btrieve and the MKDE are at last all separate entities. Things just got a whole lot simpler; the access models are totally separate and no longer depend on each other and the MKDE becomes a truly shared layer beneath those models.

One minor problem does arise from this, though. If you want to run Scalable SQL v4.0 (and the v7 MKDE beneath it) alongside Btrieve v6.15 (with its “built-in” v6 MKDE), you’ve effectively got two MKDE modules running on the same server. This isn’t allowed — neither NetWare nor Windows NT will allow two modules (or exported references) on the same machine in the same memory space. Btrieve v6.3 gets round this problem, as shown in Figure 3.

![Figure 3 - Btrieve v6.3 and Scalable SQL v4.0 in relation to the MKDE v7](image)

Btrieve v6.3?

Don’t worry, you didn’t miss out on an upgrade. Btrieve v6.3 is, effectively, v6.15 without the MKDE contained in it’s executable. When Scalable SQL v4.0 is installed onto a server where Btrieve v6.15 resides, the installation process takes the existing Btrieve engine and generates a v6.3 interface with the appropriate number of licensed Btrieve users. Note, engine becomes interface; we have to redefine our vocabulary.

Because Btrieve v6.3 uses the v7 MKDE, you automatically gain a couple of the benefits of the new version. Most notably the maximum file size rises from 4Gb to 64Gb. Another important change is in the licensing; the new Btrieve implementation, like Scalable SQL, supports “stackable user counts,” where the number of licensed users becomes far more flexible. You can, for example, take a 20-user installation and add an additional 10-user license, giving you a 30-user installation. Btrieve v6.15 has only 10, 20, 50, 100 and Unlimited users with no intermediate breaks. This means the number of upgrade products currently available (from 10 to 20 users, 10 to 50, 10 to 100, 10 to Unlimited, 20 to 50, etc., etc.) will shrink from 30 products to more like 6 — making life easier for everyone involved.

Btrieve v6.3 implements the v6.15 API, so any existing applications should behave exactly as they did with the previous version. In fact, the only indication an application will get that it is using the new version is through the B_VERSION (26) operation — which will report v6.30 for the server engine. Of course, one situation that may arise is that an application may write enough data to a file to push it over the 4Gb barrier (assuming the files are in v7 format), but this is, again, transparent to the application; the MKDE handles the “spanning” of huge files seamlessly. It’s also worth mentioning (because it is a question that has already come up a number of times in the CompuServe Forum and the Btrieve newsgroup) that applications will not have to change to handle the potential number of records that can be stored in such a huge file. Without getting too technical, the maximum number of records that can be stored in a MicroKernel file is 4,294,967,295 — which still fits into the 4-byte values returned by the B_STAT (15), B_GET_POSITION (22) and B_GET_DIRECT (23) operations.

What you don’t get with Btrieve v6.3 is the ability to use the new programming features of the v7 MKDE, such as nested transactions or temporary files. When the next version of Btrieve is released (which is rumoured to be later this year) the new features will be available through the updated API. One assumes this will be in the form of new operation codes, biases and elements currently documented as “reserved” suddenly having a purpose. There’s not enough room in this column to go further into the new features, and I’d rather not stray too far into the realm of speculation. You can be assumed that an edition of this Journal in the (we hope, not too distant) future will cover what’s new and how it’s implemented.

Summary

With the release of Scalable SQL v4.0 and Btrieve v6.3, Pervasive has moved their components a whole lot closer to the true MicroKernel architecture. ODBC isn’t covered here — and probably needs a whole column of it’s own to discuss its position in the overall picture — and only time will tell what may fall into the “Future Models” category.

Currently, Btrieve v6.3 is only available on the OS/2 Warp Server platform or as a result of installing Scalable SQL v4.0, and as far as I’m aware, no upgrade path is planned. Most developers and end users will have to wait until the next version of Btrieve is released to gain access to the new features.
ODBC

software components. If you are also using Scalable SQL, I recommend placing the Btrieve requesters in the VSSQL40\BIN directory along with the Scalable SQL requesters — then, one path or drive mapping will accommodate all client/server database processing.

16-bit applications
For 16-bit applications, it’s a good idea to copy the BTI.INI file (installed with the 16-bit workstation components) to the Windows main directory of your workstation. This file may take a little tweaking. For example, since my network was using only SPX, I had to remove the reference to TCP/IP. My BTI.INI file reads as follows:

```
[Btrieve]
Tasks=30
Local=No
Requester=Yes
Thunk=No
Verbose=0
Chkparms=No

[Communications Requester]
TCP connect timeout=15
Supported protocols=SPX
```

It seems that one of the 16-bit workstation software components creates the “Communications Requester” group when it first reads BTI.INI. The problem created at this step is that it automatically enables TCP/IP as a supported protocol. If you are not using TCP/IP, you get a status 91 (Server Error). I had to remove the TCP/IP setting of supported protocols, then reboot my workstation to make my 16-bit Btrieve application work.

32-bit applications
Configuration settings for 32-bit Btrieve applications are located in the Windows registry of the workstation. If the registry entries are not already declared, they will be created the first time you execute the ODBC driver. I had the same problems with the TCP/IP setting, however with 32-bit applications, I received a status 20 (Btrieve Not Loaded). To resolve this error, I had to remove the TCP/IP option in the registry. My registry settings are as follows:

```
HKEY_LOCAL_MACHINE/Software/Pervasive Software/Communications Requester/Version 6.15/Settings/Supported Protocol=“SPX”
```

Once your client/server Btrieve installation is up and running, installing the ODBC Drivers is a matter of making sure the proper software components are loaded on each workstation. Pervasive Software recommends using requester components released in the Btrieve v6.15.440 update (or later).

In any case, I recommend demonstrating a proper installation of client/server Btrieve by executing your existing Btrieve application, and using the Monitor Utility to verify that the files are indeed opened in client/server mode.

Step 2: Install the ODBC Workstation Components
Install the ODBC Driver from the diskette. For Windows 95 and Windows NT Workstations use the diskette labeled “ODBC Interface v2.0 for Windows.” These diskettes are bundled with all Btrieve client/server engines of version 6.15 and above.

The installation copies a number of files into the Windows main directory. The README.TXT file lists the files and where they are copied. The W32SSQL.EXE and W32SSFMT.DLL workstation software components perform the SQL parsing, local SQL buffering, and execution of the data access plan. The ODBC installation procedure does not copy the Btrieve requesters — these should have been installed in Step 1. At this point you are prompted to define an ODBC data source, however I recommend doing this from the ODBC icon in the control panel after you have made some configuration changes.

Step 3: Make sure that the configuration options for ODBC are set correctly
If you installed ODBC without a previous implementation of the ODBC driver or Scalable SQL, you shouldn’t have to make any changes. The ODBC driver will insert the proper registry and .INI settings when it is first run. Make certain that you have the following settings:

<table>
<thead>
<tr>
<th>Setup Parameters for 32-bit ODBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry Parameter</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE/Software/Pervasive Software/Scalable SQL Requester/Version 4.00/Settings/</td>
</tr>
<tr>
<td>Local=“yes”</td>
</tr>
<tr>
<td>Requester=“yes”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setup Parameters for 16-bit ODBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTI.INI setting in Windows main directory</td>
</tr>
<tr>
<td>[Scalable SQL]</td>
</tr>
<tr>
<td>local=“yes”</td>
</tr>
<tr>
<td>requester=“yes”</td>
</tr>
</tbody>
</table>

Step 4: Define DEMODATA as an ODBC datasource
The DDFs that comes with the ODBC driver are installed on your workstation. However, since you are using client/server Btrieve, these files must be copied to the server. Once this is done, you should be able to use the ODBC Administrator from the Control Panel to define these DDFs as a data source. At this point, ODBC will attempt to connect to your database. If you can connect there, you will probably be able to connect inside your ODBC applications. Now attempt to connect to the DEMODATA database from your ODBC compliant application. I was able to connect to the Patients example database table in DEMODATA.

Once you can successfully log into DEMODATA through ODBC, you can claim victory for the moment

Step 5: Log into your own database
After you have proven that ODBC is working, try to log into your own database. If you have problems doing this, at least you know that the problems are with your DDFs and not your implementation of ODBC. Contact your software developer or wait until the next issue of BDJ for a detailed discussion on Making Your DDFs Work with ODBC.

Architectural Option Three: ODBC with Workstation Btrieve
This option is required for standalone workstations. With this
option, all SQL parsing is performed at the workstation as well as all MKDE processing. Because of this, more workstation memory is required than the other two options. Network applications using this configuration will also generate an considerable amount of traffic on the network, so I recommend this option only for prototyping, standalone workstation applications, and network applications with five or fewer workstations operating with relatively light network traffic.

**Step 1: Make Sure workstation Btrieve is properly loaded on your workstation**

In many cases, when you install the ODBC Driver for the first time, workstation Btrieve v6.15 or higher is already loaded on the workstation. If workstation Btrieve is not installed, or you are using a version prior to 6.15, you have to install Btrieve v6.15 or later and the accompanying MKDE on your workstation. Workstation Btrieve and the MKDE is included with Workstation Btrieve Developer kits, singleseat copies of workstation Btrieve, and Btrieve Commercial Deployment Kits.

I recommend placing all the Btrieve workstation software components in one directory, then setting a PATH to that directory. The developer kit, by default, installs the 32-bit software in the C:\Program files\BTI\WIN32\BIN directory. The W32MKDE.EXE file is the workstation version of the MKDE. Your workstation must be able to find this file and its associated files including WBTRV32.DLL — the 32-bit requester. 16-bit workstation software components are installed in the C:\BTI\WIN\BIN directory by default.

I recommend demonstrating a proper installation of workstation Btrieve by executing your existing Btrieve application before you attempt to make ODBC work. If you don’t have an accompanying Btrieve application, use the Btrieve File Manager Utility (WBMANAGE.EXE, a 16-bit application) and open your data file—this will make sure that Btrieve can read your file.

**Step 2: Install the ODBC Workstation Components**

I recommend a fresh install of the ODBC driver on your workstation. If you have used Btrieve or Scalable SQL in the past, your registry may have parameter entries that the ODBC driver uses. If you delete these registry entries before installing ODBC, you can be assured that there aren’t any “pre-existing conditions” with your configuration settings. The ODBC driver and Btrieve will automatically create the necessary registry entries if they are not already defined. The same is true for 16-bit applications and the BTI.INI file.

Keep in mind, however, if your workstation is also performing client/server processing with Btrieve or Scalable SQL, deleting the configuration options may adversely effect existing applications. That is one reason why I recommend this type of architectural option — ODBC with workstation Btrieve—for only stand alone workstations. If your workstation is currently using Client/Server Btrieve 6.15 or above or, Scalable SQL v4.0 or above, you may wish to consider running ODBC in a client/server configuration—remember though, in order for ODBC to work in client/server mode, your data files must be located on the file server, not the workstation.

Use REGEDIT to delete the following registry entries.

```
HKEY_LOCAL_MACHINE/Software/Pervasive Software
HKEY_LOCAL_MACHINE/Software/Btrieve Technologies
```

**Step 3: Make sure that the configuration options for ODBC are set correctly**

If you delete the above registry entries, your configuration options will be properly initialized for stand-alone workstation processing. This will happen when you first run the ODBC Administrator and your ODBC application. My registry contained the following settings.

```
  Setting | Value
  -----:|------:
  Requester | yes
  Local | yes
```

**Step 4: Define DEMODATA as an ODBC datasource**

The DDFs that come with the ODBC driver are installed on your workstation. Use the ODBC Administrator from the control panel to define these DDFs as a data source. At this point, ODBC will attempt to connect to the database.

You might encounter memory-related errors when you first try to connect to a database. Remember, this configuration is the most memory intensive of the three options and you may experience memory-related lock-ups with your ODBC application. You should probably close other applications that are running on your workstation. In addition, make sure you have sufficient disk space available. My workstation had eight megabytes of available disk space and 16 megabytes of RAM.

**Step 5: Log into your own database**

Once you can successfully log into DEMODATA through ODBC, try to log into your own database. If you have problems doing this, you may have problems with your DDFs or you may be running out of memory. Contact your software developer or wait until the next issue of BDJ for a detailed discussion on Making Your DDFs Work with ODBC.
Smithware Crystal Reports 5.0 for Btrieve

By Craig M. Bobchin
cbobchin@csi.com

All database applications need reports to condense and make the volumes of data in them understandable. While many systems come with their own report writers, there is always room for improvement. That is why third party reporting tools first came into being; starting out as replacements for the very limited and slow integrated report writers.

Crystal Reports is the current champion of report writers on the Microsoft Windows platforms. The latest version, 5.0, adds a number of new features, including multi-pass reporting, new export options like HTML, and sub-reports. Available in both 16-bit and 32-bit editions, it will run on any Microsoft Windows platform.

Data, Data Anywhere

One of Crystal’s biggest selling points is its access to virtually any data that can be reported on. Smithware Crystal Reports for Btrieve comes with native drivers for the following formats: SQL Server (Both Sybase and Microsoft), Oracle, Informix, Btrieve (Smithware includes an enhanced set of drivers for Btrieve), Paradox and most other PC database formats. Smithware Crystal Reports for Btrieve can also be used to run reports on data located in Windows NT event logs and in Microsoft Exchange. Additionally, Smithware Crystal Reports for Btrieve has ODBC access so you can get to any data that there is not a native driver for as long as you can create an ODBC data source for the data.

There are some advantages and disadvantages to using the native drivers instead of ODBC to access your data.

Using native drivers to access the data is the fastest route to reading the data. In this case, Crystal Reports only needs to communicate with a single data access layer. Also data access is simplified; native drivers are treated as point-and-click data sources. Just select the database files and Crystal Reports will read the stored data.

Among the cons of using native drivers, in accessing a database directly from Crystal Reports, you will find that only one database type can be used by the report. You will not be able to switch to a different database format or even a different table without creating a new report.

The reason is that the native drivers communicate with the database data using database-specific syntax. This means that Btrieve data access is not the same as dBase data access. If on the other hand you use ODBC, the syntax is always SQL no matter what the actual database type is.

Installation

Installing Smithware Crystal Reports for Btrieve is an easy task and will take about 30 minutes. In addition to installing the Crystal Reports system, it will also install an assortment of utilities that will make life easier, including DDF builder and DDF Sniffer, and Smithware Navigator, which acts as a menu system to the entire package. These programs are very useful if you need to build and examine Btrieve data files.

The whole installation will eat up over 50 megabytes of hard drive space. One glaring problem I noticed in the installation is that there is no way of selecting what optional utilities get installed. In my case all of them got installed whether I wanted them or not. You can install the 16-bit version or the 32-bit version of Smithware Crystal Reports for Btrieve or both, however you have to run through a separate installation for each version.

After you install the program I suggest you check the Smithware web site http://www.smithware.com to see if any patches have been posted. There was at least one patch available at the time of this writing.

Report Creation

Once you get the program installed, it is time to start creating a report. Here is where it gets interesting. Smithware Crystal Reports for Btrieve gives several options for you to choose from in creating a report. When you first start the program you are presented with a welcome dialog that asks whether you want to create a new report or open an existing report. If you choose a new report, you are presented with a report gallery in which you can select from a dozen choices. Eight of the choices will bring up an expert or wizard to help you create the report.

The experts give you a choice of Standard, Crosstab, Form, Drilldown, Form letter, mailing label, top N or Sub-report. Sub-reporting is one of the new features of Crystal 5.0. There are also several other report types you can create, but they do not use the experts. If you use one of these other ones you will have to set the datasource when you create the report. I’ll look at creating a standard report with the report expert first.

In the report gallery, click on the standard report icon. This will bring up a tabbed and buttoned dialog box that walks you through the process of creating a report. The first step is to create or choose a data-source. Here Crystal gives quite a few choices for data sources. You can use ODBC/SQL data, one of Crystal’s native driver formats. A crystal query or dictionary (these are created with the Crystal Query and Dictionary builders respectively), or an Oracle database, Web log, or NT Event log.

For our purposes we will use a Btrieve table and get it via the native driver. By choosing the Datafile button and then the File:DDF file we get a list of all tables in the Btrieve database. The next step is to create relationships between the tables. Smithware Crystal Reports for Btrieve has a function known as “Smart Linking” that simply looks at indexed fields with the same name between tables and links them. You can also link tables explicitly by dragging and dropping between the linked fields.

This Smart linking suffers from a couple of problems, first and foremost is the size of the screen you have to work with. It is painfully small. You can fit three tables across but only one or two down before you have to scroll. And believe me if you have to link several tables you will scroll, and will soon get tired of it. The other problem is with the Arrange Button. If you move the tables in the display and then press this button, you may have to rearrange the tables to suit you. Half the time it seems to put the tables back in the original order the other half it did nothing at all. Once you have the tables linked it is time to choose the fields. Fortunately this is a lot more straightforward operation.

The Fields tab gives you a list of the tables in your report and their fields. By double-clicking on a table you can expand it to show all of the fields or shrink it so all you see is the table name. You can single click on the fields to select a group of them the click the Add button or double click a field to move it to the report. You can also set
the column heading of a field and create calculated fields (called Formulas) that can appear in the report here. We’ll get into formula fields later.

Smithware Crystal Reports for Btrieve also gives you the option of browsing the data in a particular field so you can see if it is really what you want in your report. The next step is to add a sort order to the report and then any totals you may want. You can do this through the expert tabs, but I would suggest previewing the report and then adding the breaks, sorting and totals from within the report preview window. One problem I noticed with the expert is the lack of a “Finish” button. For users accustomed to other products with similar functionality this can be a bit confusing.

Both the Design and Preview tabs are bound together. When you make a change in one, that change is made in the other.

The Preview Tab identifies report sections in the shaded area to the left of the data. With a quick look you can tell which report section the data is printing from. While section names appeared only once in the Design Tab, they print each time section prints in the Preview Tab.

The Preview tab has a few enhancements that the Design tab lacks thereby helping to set them apart.

For one thing in the Preview tab you work with the actual data, rather then a field frame representing many field values.

In the Preview tab, you will find that the record counter, the data age indicator, and the page movement controls are active.

When you select a field in the Preview tab, Smithware Crystal Reports for Btrieve highlights every value that appears in the report. In the Design tab only the field frame is highlighted.

The Preview tab has one vertical ruler on the left side of the tab. The Design tab has individual rulers for each report section. In both Preview and Design modes the rulers behaves the same.

The process of building and modifying a report is the same in both the Design tab and the Preview tab. You should find it easy to work with your reports in both places. However, if you do work in the Preview mode you should be aware that if you are working with a large data set actions such as moving and reformating fields may slow down.

Integrating the Report and Your Application

Once you have your report developed, it is time to integrate with your application and get it to the end-users. Crystal Reports gives you multiple options to use in distributing your report. You can use the Crystal Report custom control, an ActiveX control, a Windows DLL file callable via an API, a VCL for Delphi and even the option of compiling your report and the runtime engine into a standalone EXE file.

I suggest using the ActiveX control. It gives a lot of control and is easy to use. If you need more control, you can always use the report engine DLL. It includes a well-documented set of API calls that can be used to fully customize your reporting needs.

Documentation and Help

Smithware Crystal Reports for Btrieve ships with two manuals for Crystal Reports, One from Seagate Software that covers general reporting issues, and one from Smithware that goes into detail about Btrieve and how to work with Btrieve data and Crystal Reports. It also explains how to use the Btrieve Client and DDF Sniffer programs that comes with the package. There is a separate manual for DDF Builder.

The documentation was good, with adequate screen prints explaining how to use the programs. There are also nine help files covering all aspects of the program. The help files do a good job of filling in the gaps in the printed documentation. For example, there is minimal printed documentation about using Crystal Reports in your applications. There is a large help file that specifically addresses developers needs. However, there are not a lot of examples for Visual Basic.

Conclusions

I like Crystal Reports. Version 5.0 has made great strides over previous versions. It is now easy to add sub-reports to a report; multiple section reports are now possible as well. You can integrate your report with your applications via an OCX control, standard DLL API calls, or as a compiled report. Coupled with the ability to read data from virtually any data source you can think of this is one powerful reporting tool.
install wbtrv32.dll. This effectively disrupts the operation of Btrieve for any application. The file needs to be renamed after the installation of the distributed report if you get the error.

**Using the Btrieve API in a C++ environment under Windows NT 4.0 results in the error message “External Exception at C0000008”**

This is a bug in the Btrieve engine, and is very easy to solve since Pervasive has a patch currently available.

**Solution:**
2. Run BTRW32CE.EXE to unpack patch software.
3. Run PATW32CE.BAT. This applies the patch.
4. Copy W32*.* and WBTR*.* from WINDOWS\SYSTEM to WINDOWS\SYSTEM32.

**Note:**
The patch assumes that you’re running with Windows 95. With Windows NT, you also have a SYSTEM32 directory, and it’s from there that applications attempt to call the Btrieve DLLs.

**ActiveX Controls for Btrieve: “run-time only” properties of the controls do not appear in the Class Wizard in Microsoft Visual C++ version 5.0**

When a Smithware ActiveX Controls for Btrieve custom control is added to a Microsoft Visual C++ version 5.0 project, Class Wizard fails to create source code templates for any properties of the control which are available only at run time. This is not the case with any previous version of Microsoft Visual C++, which will create the appropriate classes with template code for both “run-time” and “design-time” properties. Smithware has made Microsoft Class Wizard-generated template source code available on the technical support web site in the Updates and Patches section, http://www.smithware.com/support/patches.html, under the file name classdef.zip. Replace the Microsoft Visual C++ 5.0 Class Wizard-generated files with these templates to include reference to “run time only” properties in your project source.

**ActiveX Controls, extended operations and Btrieve status 62**

Btrieve will return a status 62, invalid extended operations descriptor error, on GetNextExtended, GetPreviousExtended, StepNextExtended, or StepPreviousExtended methods of the VAccess data control if the SelectedRecords property of the control specifies a comparison a field defined as any one of the Smithware extended data types. Extended data types are listed on pages 95-96 of the DDF Builder for Windows Reference Manual.

Several of these data types, (those designated with an asterisk beside the FIELD.DDF type code) cannot be used in extended operations SelectedRecords comparisons, since Btrieve has no key type which will collate the stored data value correctly. Using this “placeholder” FIELD.DDF type code 0 (string) as the data type for the comparison field, the operation would proceed without error, but the results of the SelectedRecords comparison would not be meaningful.
v = VAccess1.FormatFieldValue ("total_amount", ",##,##0.00")

**Patches and Updates** - A file listing for each component of the patch- es can be found at the Smithware Tech Support Site

www.smithware.com/support

- cbldfix.zip (8K)
  Workaround for ActiveX methods giving compile time errors when invoked in Borland C Builder++. See readme file for instructions. (PKZIP format).

- classdef.zip (7K)
  Class for the Microsoft Visual C++ version 5.0 environment that contains both the runtime and design time properties. (PKZIP format).

**Sample Applications** - A file listing for all of the sample apps can be found at the Smithware Tech Support Site

www.smithware.com/support

- csample2.zip (62K)
  Sample code for Smithware ActiveX Controls for Btrieve. It demonstrates using the VAccess control in a C++ Doc/View application, without using the resource editor. It should be unzipped using the -d option, to maintain the directory structure. (PKZIP format).

- cbldsamp.zip (177K)
  Sample code for Smithware ActiveX Controls for Btrieve. It demonstrates using the ActiveX Controls in Borland C++ Builder. (PKZIP format).

**How to Take Advantage of Smithware’s Technical Support Resources**

At Smithware, we make every effort to provide free, fast, quality tech support to all of our registered customers. Over the past months the demand for tech support has grown dramatically due to the introduction of new products and increased sales volume. We have added more staff to the support department to help accommodate this incoming load. There are ways that you can help us speed up our response time and get your questions resolved quickly.

1. Check the manual or online help first.
2. Search our online knowledge base at www.smithware.com/support. It’s fast, fun, absolutely free, and available whenever you need it. Our online knowledge base contains answers to the most commonly asked questions, helpful hints, and other useful information.
3. Send in a fax or an e-mail with your name, company, phone number, e-mail address and fax number, and as detailed a description of the problem as you can provide. Please include which product and version you are asking about, the text of any error messages, what operating system you are running, what Btrieve engine versions are involved and any other details you think may be helpful. A form for submitting technical support requests via e-mail is available on the Smithware technical support web site.
4. Smithware technical support representatives are available to assist you from 8:00 a.m. to 5:00 p.m., Central time, Monday through Friday.

**Contacting Tech Support**

Remember that you must be registered to be entitled to technical support. So please send in your product registration card if you purchased your copy through one of our distributors. If the product was purchased directly form us then we will automatically register you. If you have a Smithware version of Crystal Reports then it must be registered with both Seagate Software and Smithware.

e-mail: support@smithware.com
http://www.smithware.com/support
CompuServe: 75470,546
Telephone: 615-386-3100x71
Fax: 615-386-3135