The Value of Consulting Services

As a consultant, I'm keenly aware of the bitterness some people have toward us. On many occasions I’ve felt the ire of managers who, in the past, have had bad experiences. The causes vary, but the blame can usually be spread evenly between management and the consultants. Unfortunately, this results in a negative view toward consultants and the valuable services they provide.

At the risk of stating the obvious, let me describe some important details about the business of consulting:

1. CONSULTANTS PROVIDE VALUABLE SERVICES.

When you decide to build a home do you build it yourself or hire a building contractor? If your car breaks down do you hire a mechanic? Why don’t you take care of these tasks yourself? The answer is that you probably don’t have the skills or the time. You could take the time to learn, but usually it’s better to hire someone else. The same goes for consulting services.

If you need extra help on your project, you probably can find a consultant who can help you. Though it may cost a lot of money, it’s usually far cheaper than having your staff learn the skills on their own. In addition, you can use the consultant to fix your problem as well as train your staff in that skill area. Don’t be afraid to hire consultants to help you out. If for no other reason, it’s a good idea to bring one in to get an outside opinion regarding your new design or technology decisions. Some companies even have a policy of including outside review prior to making big architectural decisions.

2. IT’S REASONABLE FOR CONSULTING RATES TO BE ABOUT THREE TIMES THE HOURLY RATE OF A FULL-TIME EMPLOYEE WITH THE SAME SKILL SET.

Being a consultant is risky business. First, if you don’t find steady work, you don’t have income. Even when you do have work, it isn’t likely you’ll be billed 40+ hours per week. Therefore, your rate has to be high enough for you to live on. Second, independent consultants, and even those who work for consulting companies, often have to pay for their own insurance, payroll taxes, and other business costs. The other big factor in the high cost is the middleman.

Whether it’s a consulting company, temp agency, or recruiter, someone brings the customer and the programmer together. That someone gets a cut of the hourly wage, boosting the billed rate either a little or a lot. Rather than think poorly of your consultant for the high rate you’re paying, consider his or her costs and risks. You might discover that the rate isn’t so bad after all.

3. RECRUITERS AREN’T EVIL…BUT AREN’T ALWAYS NECESSARY EITHER.

As with any industry with a demand and a supply, there needs to be some method to bring buyers and sellers together. In our industry, recruiting and consulting companies do that work. Of course, as middlemen, they take a cut of the pay. Sometimes it’s as little as $10 per hour but often it’s as high as $50. The high cut is one reason they have a bad reputation. However, they perform an important task that’s often indispensable. If you use a recruiting or consulting company, don’t be afraid to ask what cut it’s taking. If it’s overcharging, the company won’t want to discuss that with you. In that case consider doing business with a different recruiting or consulting company. I know an expert-level developer who was paid $65 per hour as a nonbenefit hourly employee but was billed at $115 per hour. That customer was being squeezed by the consulting company for $100,000 per year.
When I returned to the office the next day, my enthusiasm was greeted with equally proportional apathy. PowerBuilder 4 applications were still being built. Using 5.0 wouldn't be considered for some time. Going to a distributed architecture was a dream. Those days are long gone. DPB has been replaced with Jaguar CTS and ASE. PowerBuilder 7 has been in GA long enough for several point releases to come out and for PowerBuilder 8 to be debuted at TechWave. Sybase's solution for distributed architecture is proven and reliable. So what are you waiting for? Probably the same things I'm waiting for: a budget, time, customer buy-in, management buy-in, a usable infrastructure, to name a few. Having to wait for permission to employ the best solutions has become an axiom to developers throughout the world.

This article will show how to build client/server applications ready to be distributed using Jaguar CTS. I know what you're thinking. “Yeah, yeah. Use NVOs. Make sure I've checked 'Jaguar Validation' in the User Object painter. I've heard all that before.”

While I do need to mention these points to convey accurate instructions, they won't be the crux of this article. I’ll explain how to build your systems so you can deploy them as client/server and then, whenever you're ready, fire up Jaguar CTS, register your objects, create your proxies, and not rewrite a lick of code. Yes, you'll be able to distribute your application with a flick of the switch. Not only that, but if for some unforeseen reason you need to disconnect your Jaguar server, you can switch everyone back to local PBDs.

Several aspects of distributing your application with Jaguar must be clearly understood before you embark on a design such as the one I’ll be discussing. This article assumes that you already know and understand the basic concepts of Jaguar and its interaction with PowerBuilder 7. This doesn't serve as a lesson in Jaguar. If you find this article difficult to understand, I recommend you study Michael Barlot-
Local vs Remote Instantiation

A nondistributed application instantiates objects locally, that is to say, in client memory. Your PBDs usually reside on the user's machine or on a shared network drive. A distributed application instantiates objects on the Jaguar server, and the client utilizes their methods via stubs, or proxies. Once an object is instantiated, however, PowerBuilder doesn't care where it actually resides. Its location is transparent. That doesn't mean you can take your objects, distribute them, and all will be well. More than likely, a number of scripts will require recoding with reference to how your objects are loaded and used.

Herein lies the challenge of converting your application to be distributed. It's not just a simple process of registering your objects in Jaguar and generating the necessary proxy objects. Consider this: to instantiate an object locally, you code the CREATE statement. To instantiate an object remotely, you need to call the Create-Instance method of the connection object. Then, of course, there's the destruction versus deactivation of the object. Think about how many places in your application you instantiate and subsequently destroy NVOs. Now think about recoding all that to act against a Jaguar server. It hurts, doesn't it?

Fortunately, PowerBuilder supports the object-oriented concept of polymorphism, natively referred to as overloading. By properly overloading key methods, your application will be smart enough to know where to instantiate its objects and what connection methods to use. While this concept is straightforward in theory, it takes good planning and self-discipline to stick to your designs to make it work.

There are also restrictions inherent to Jaguar that will force you to change your way of coding. For instance, if you code using a lot of loose coupling between objects via DYNAMIC method calls, you'll have to start using STATIC calls, since Jaguar doesn't support DYNAMIC. (Your code will compile a DYNAMIC method call to a Jaguar component, but it will generate a runtime error stating the method wasn't found.) You'll also need to clearly identify which methods will be PUBLIC, PROTECTED, and PRIVATE. Jaguar restricts the datatypes that methods may use as arguments as well as the types they can return to the client, thereby necessitating that you identify all potentially PUBLIC methods. For instance, I've always followed the model of starting with PRIVATE as the method access, changing it to PROTECTED or PUBLIC only as application development dictates the need for access. If I have a method that uses the Any datatype as an argument or return, I can't make it PUBLIC and still use it with Jaguar. All PowerBuilder datatypes are supported in PROTECTED and PRIVATE methods, but some (such as Any) will fail when you try to compile your method as PUBLIC.

As we move along in this article, I'll point out other restrictions as needed. For now, let's look at how to make your application smart enough to know whether it's distributed and then how to instantiate its business objects (NVOs).

### What we really want is a custom Connection method to handle the instantiation and destruction/deactivation of any NVO regardless of its location

When your application starts, more than likely it will require an immediate connection to either your SQL database or the Jaguar server. How does it know where to connect? In the example we'll be discussing, the application will use an INI entry, "ConnectTo=", which will specify either "JAGUARCTS" or "SQLSERVER". (You could store this entry in a registry or even a login window. Just remember that it's the "switch" we'll be throwing later, so it should be somewhere it can be easily modified. Don't place it in a script!)

Based on this entry, the application manager object, n_pbdj_appmanager, will know how and where to connect. The Constructor event of the application manager posts the of_Connect() method (see Listing 1), which calls the appropriate connection method, of_ConnectToServer (see Listing 2) or of_ConnectToJaguar (see Listing 3). The SQL Server connection method does a run-of-the-mill SQLCA connection followed by an instance variable, ib_isDistributed, set to FALSE. The Jaguar connection method performs the necessary steps to connect to the Jaguar CTS server, and upon a successful connection the ib_isDistributed variable is set to TRUE. Essentially, once you've connected to the appropriate server, you're considered either distributed or local.

For the remainder of the application's processing, a public method of the application manager, of_IsDistributed(), will return the value of ib_isDistributed to indicate how object instantiation and data manipulation should occur.

### Where to Connect

In Listing 1, the custom Connection method performs the necessary steps to connect to the appropriate server, n_pbdj_appmanager, will know how and where to connect. The Constructor event of the application manager posts the of_Connect() method (see Listing 1), which calls the appropriate connection method, of_ConnectToServer (see Listing 2) or of_ConnectToJaguar (see Listing 3). The SQL Server connection method does a run-of-the-mill SQLCA connection followed by an instance variable, ib_isDistributed, set to FALSE. The Jaguar connection method performs the necessary steps to connect to the Jaguar CTS server, and upon a successful connection the ib_isDistributed variable is set to TRUE. Essentially, once you've connected to the appropriate server, you're considered either distributed or local.

For the remainder of the application's processing, a public method of the application manager, of_IsDistributed(), will return the value of ib_isDistributed to indicate how object instantiation and data manipulation should occur.

### How TO USE THEM

CreateInstance, as you know, is a method of the Connection object. Therefore, that's where we want to handle all our NVO instantiation and destruction, even if we're not connected to a Jaguar server. Our connection object, n_pbdj_connection, has the overloaded method of_CreateInstance shown in Listings 5 and 6. To use the method, declare your NVO variable as you normally would. Variable scope isn't an issue here. The following code snippet shows how to use of_CreateInstance to instantiate an instance variable of type n_pbdj_math in our window.

```powerscript
/* Clicked Event of cbInstantiate command-button on w_pbdj_main window */
IF gnv_connect of CreateInstance( n_pbdj_math, "PBDJ\". n_pbdj_math ) = 0 THEN
  sle_object.Text = "n_pbdj_math instanti-ated"
ELSE
  sle_object.Text = "Instantiation failed."
END IF
/* End of Clicked */
```

The of_CreateInstance method accepts three arguments. The first, n_pbdj_math, is a reference to the variable you're instantiating. The other two are string value arguments, avs_package and avs_component. If the application is distributed, that is, connected to Jaguar, the arguments are used to construct the
class name argument of the Connection object's native method, CreateInstance. If the application isn't distributed, the avs_component argument is used to instantiate the arn_object argument using the CREATE USING syntax. (As you know, CREATE can be used with USING to specify a specific descendant class other than the declared type.) Notice how the component name is the same as the NVO name. This is so the method can instantiate the object without caring about its location. This is a very important aspect of this design that must be followed for it to work.

When you generate your components on Jaguar and the related proxies in your PBL, the name of the NVO as a local object must be the same as the proxy/component name. Otherwise, the of_CreateInstance may attempt to instantiate a nonexistent object. Also, do not generate your proxies with the "Prepend Jaguar package name to object name" option checked. This causes a variable-type mismatch, and the instantiation will fail on one of the two types (remote or local).

Another important piece of this design is the use of the avs_package argument. You must clearly identify what package names will be used when you distribute the application. While you run the application locally, this argument is ignored. But once you "flick the switch," your component must exist in the package specified in your argument. Of course, if you choose to use only a default package, you can use the overloaded of_CreateInstance method that doesn't use the package name (see Listing 5).

Once you're finished with your object/component, call the of_DestroyInstance method as shown:

```c
/* DESTROY arn_object */
arn_object.of_DestroyInstance( ) // Ancestor method
ELSE
    MessageBox( "Connect To Jaguar", "Unable to connect to Jaguar. Code : " + String( ll_rc ), Exclamation!
    RETURN -1
END IF
```

Listing 7

Testing the Theory

To test this approach you'll need to have your Jaguar component's script out of sync with your NVO. Let me explain why. I have a method in n_pbdj_math called of_Add. All it
does is accepts two numbers, adds them together, and returns the result. In my test window I enter the two numbers and get a result. If I'm connected to Jaguar and using a remotely instantiated object, one plus one returns two. If I'm connected directly to SQL Server and using a locally instantiated object, one plus one returns two. Unless I feel like doing all my testing via the debugger, I need a better test. So you need to modify the NVO itself (after you've already deployed it to Jaguar) to return, let's say, argument one plus argument two plus 10. Now your component on Jaguar returns a correct value for any two numbers you add together, while the local object always returns an incorrect value (correct results plus 10).

The test window I use (see Figure 1) allows me to choose whether I'm distributed or not, instantiate and destroy the math object as needed, fire its four methods, and see their results. When I switch to Jaguar, my of_Add method always returns the correct result. When I switch to SQL Server, I'm always off by 10. The switch works! Now I can tell my application to run locally or remotely with one “flick of the switch.”

That's Not All, Folks!

You can imagine the value of using this approach in building new client/server applications. Sure, you'll still have a lot of work when it comes time to distribute your system. Aside from having to set up the Jaguar servers and clusters, you'll need to load your components into Jaguar, generate the proxies, and redistribute your PBD files to the appropriate machines. But you won't need to rewrite any instantiation/destruction code! That's reason to party!

And that's not all. If this were as far as the solution took you, all your business objects would be distributed but you'd still require a database connection for your DataWindows, DataStores, and embedded SQL. You'd eventually bog the network with open database and Jaguar connections instead of letting Jaguar handle pooled database connections for you. That would defeat the purpose of setting up your Jaguar servers and clusters. To distribute your system with a flick of the switch and take advantage of the benefits therein, we need to code our database data manipulation using a similar, but much stricter, design. That'll be discussed in my next article.

Author Bio

Tom Peters is a project leader at Incremax Technologies Corp. in New York City. A CPD professional with 14 years of IT experience, he teaches regularly at the Sybase TechWave conferences.

tomp@incremax.com
Data Mining - Discover Your Hidden Treasures

What to dig for

You may think you’re using your data warehouse to its fullest extent. After all, you’ve amassed a great deal of data about customers or inventory that you can look up at will. But if you’re not using the proper business intelligence (BI) tools, your data warehouse is no better than a hunk of coal. BI tools such as data mining can turn the coal into a diamond that can propel your business to the top.

Data mining can help you do more than you ever imagined with your data warehouse. Some call data mining data discovery or data knowledge, but whatever you call it, data mining allows you to take your data warehouse, dissect it piece by piece, and summarize it into useful information.

Data mining isn’t just a fancy term for OLAP (Online Analytical Processing), which tells you what happened in the past. Data mining helps users predict the future through the discovery of trends and patterns in the data. This insight can then help a business repeat good trends and avoid bad ones. Most businesses use a combination of data mining and OLAP. It’s important to keep the difference in mind so the proper tool is used to answer the question at hand.

Marketing Departments Love It

Data mining tools are especially important for marketing departments because of the need to single out customer segments with high profit potential. Before, it was difficult to identify these groups because it required manually sifting through large amounts of data. Data mining makes it quick and easy to sort through the data and discover customer behavior and patterns.

For example, if a company is examining the effectiveness of its advertising campaign and wants to find out why people are coming to its Web site and who’s actually buying products, data mining tools can help. The company can sort through its data warehouse to profile customers concerning which referring Web sites generate sales and which don’t. This helps the company better target future marketing efforts.

Choice of Tools

A number of data mining tools are available. Some of the more popular are Cognos Scenario, Angoss Knowledge Suite, Business Objects Business Miner, SAS Enterprise Miner, and SPSS Clementine. Microsoft SQL Server 2000, due out in a few months, will have data mining features. There are also related tools that do predictive modeling or present the results of the data mining graphically. But which program is best?

The best one for you may not be best for another company. It really depends on the level of management and sophistication you need. If you’re a visual person who needs to see it in a graph, you want something like Cognos Visualizer. If you’re a person with a bunch of thoughts, but not sure where they lead, you may want to use Cognos Forethought.

To use data mining tools you need to clean up your data warehouse and integrate it into one database. Then figure out which type of data you’re going to use. Once you’ve gotten your core listing of data cleaned and ready, you need to transform it into the appropriate form for use by the mining tool.

When you start mining, use a pattern finder to figure out what types of patterns exist in your data. Once you find some patterns, you need to figure out which are useful and interesting and which aren’t. This may take some time.

Get the Most from Your Data

The use of data mining is still up and coming, but it’s becoming clear that without data mining tools, businesses aren’t getting all they can out of their data. The growth of e-commerce has increased the need for data mining because Web-based companies are amassing huge amounts of data on their customers, which would be impossible to analyze with traditional tools.

Remember to keep the key decision makers involved during the entire data collection and mining process as they’re the ones who’ll need to know how best to use it. Think of decision makers as the miners with the helmets and lights. You can send them into a cave, but if you don’t tell them what to dig for and how to do it, they won’t find anything.
In the October issue of PowerBuilder Journal, we worked on the design for our mail server on Jaguar. This is more accurately called an MDA, or Mail Delivery Agent. What we’re writing isn’t the application that will receive raw messages from the Internet, parse them out, then store them for another application to download. It in fact does the downloading.

This relationship is beyond the scope of this series, but it needs to be addressed simply to avoid confusion. What we’re considering is something far closer to a Microsoft Outlook than a SendMail.

So why write a replacement for Microsoft Outlook? Actually I’m a big fan of that product. I use it daily. We’re doing this because it’s a fairly straightforward exercise in building Jaguar components and is in fact about as purely distributed as anything can be. During the course of this article we’ll find more tangible reasons to write this application, but for now just consider it an exercise.

So let’s get back to our work here. It’s been a while since the first article so let’s review. Figure 1 came from that article. Let’s look at it again.

What’s most important to this article is the middle objects, the mail server and the mail customer. In the previous article we got started on the mail server. We created stubbed-out methods for getting headers, logging on and off, saving, and sending messages. We’re going to have to flesh this out.

This is where I reveal a little secret about object-oriented design. There’s a concept in OO technology called stochastic animism. To oversimplify this issue, each object should in some way model an object in the real world. It’s distinguished by the question: What does this object know how to do?

In our case we want two distinct sets of functionality. We want an object that handles the connection to our MAPI. This would be our mail customer, who knows how to send and retrieve mail. Basically he’s our interface to the MAPI functions.

The MailServer component is tasked with gathering this information and returning it to the user the way we want it. He presents the information to the user. He handles the HTML. If we decide to enhance our application down the road, it would be he who communicates with other objects to bring the entire application to a cohesive front for the user. Suppose we want to add e-mail lists to our application. We’d create a list manager component and reference it from our server. Our user would communicate with the server to ask to send e-mail to a list. Our server would combine the list manager and the mail customer to handle that request and return HTML to our user, each object doing only what it knows how to do and nothing more.

This is the essence of stochastic animism. It’s served me well in my career and has greatly simplified the design of objects for me. I’ll follow this methodology in our series here.

Now that that’s settled, in my previous article we created stubs for several functions. Now we’re going to use them. The first thing we need to do is create our mail customer component. The mail server will use it to process those actions that the user requires. The mail customer component will never be used by the end user, only by the mail server. That means – and this is important – we don’t have to be CORBA compliant with it. In other words, we can pass and return any kind of variable that PowerBuilder understands, not just the primitive variables that CORBA understands. This gives us a lot more flexibility.

So let’s create the component. You’ll click on New, then make sure you’re in the Object tab page. Double-click on Jaguar Component Wizard – just as in Figure 2. It’s important that you choose this one. There are two other places where you can create a Jaguar Component: in the Start Wizards and in the Projects tab pages. You want a component added to this project, though, so this is the appropriate selection.

I’m not going to step you through all the dialogs of this wizard. Just select the defaults until you get to “Specify New Component.” The default for that will be the component you already have. Change it to your new component name. I changed mine from the default “n_jag-MailServer” to “n_jagMailCustomer”.

Eventually you’ll have to select a package. Choose the one that contains your mail server.

You’ll be asked if this is a Standard, Shared, or Service Component; it’s Standard.

In the Specify Transaction Support you’ll need to make sure that the Auto Demarcation/Deactivation remains off. This is a deviation from my normal components. It means that the component will be stateful and remain in memory from one method call to the next. If we checked this box, it would be stateless and we’d have to reconnect every time we wanted to call a method. I always make components that will communicate with the user stateless, but since I
have full control of this component from my mail server, I feel safe in making it stateful. In another article we'll create functionality to deal with time-outs and deactivating these components.

In the Specify Other Component Options we want to Support Remote Debugging. Whether you support live editing (automatically deploys the component every time you save it) is up to you. I find it annoying so I don't do it. It does mean that I need a separate project for it to deploy the component, but I get annoyed when I make a tiny change, then have to wait for the component to deploy when I save it.

In the Specify Dynamic Library Options you should check the box that lets you include unreferenced objects in the consolidated PBD. This way I don't have to worry about accidentally forgetting to put something into a PBR file.

Now we have our object. Open it in a painter and let's get started with it. First you want a mail session as an instance variable. The definition is:

```java
private MailSession ms_mailsession
```

You now want to create a logon method that will be called from the mail server. Basically, the mail server is going to get a request from the user to log on. He'll create a copy of your mail customer component and call the logon method of that. He'll then translate the return value to a CORB-compliant return value and return it. You now need a method called of_logon that will return a MailReturnCode and take a name, password, and mailLoginOption as arguments. This is just one line of code so I'll add it right here:

```java
return ms_mailsession.maillogon(as_username, as_password, ao_option)
```

Now we have a method that our server can call. That brings us back to the issue of the relationship between the server and the customer. We already settled that the server would handle multiple customers. Clearly that means an array of customer components. However, we need to keep track of whether a component is active. Look at it this way: we'll create these customer components in our server and do things with them. At some point we'll no longer need them. We can either return them or hold on to them and reconnect them at a later date. I like the idea of holding on to them. To handle that we need another instance variable in our customer to tell us if the component is logged on. Add this line of code to the instance variables:

```java
private boolean ib_connected = FALSE
```

By default, all instance variables are public. In this case I want my server to be able to read the variable directly, so I can leave the Read attribute public. I don't want my server to be able to change the value though. I want only this particular object to be able to change it and do so only in response to a successful logon. That explains the privateWrite.

I now have to modify my of_logon method to set this instance variable. Oh, well, that's how component design works. Go back to your of_logon and change it so that instead of the one line, it now matches Listing 1.

### Back to Our Server

Now we have something we can call. We can log on (hopefully). Let's go back to our n_jagMailServer and create that line between the customer and server shown in Figure 1.

The first thing we need is an array of customers. We should make that an instance variable. Go to the instance variables of n_jagMailServer and add:

```java
private n_jagMailCustomer io_customers[]
```

We need to write a little code to handle this array. It's a simple method that will loop through the array looking for customers that aren't connected. These are components that were connected once but have finished their tasks and are waiting to be reused. If it finds one, it will have to reuse that array element. It won't need to create a new one. I call this method of_findAHole. It returns the array index to an unused element or creates another customer and adds it to the last array element, returning that index.

The code for this method is in Listing 2. We finally get back to our of_logon in the server. It takes three strings as arguments and returns an integer. This doesn't match our customer logon. We need to translate. The mailOption argument passed in is fairly simple. Let's just code that right in the method. Translating the return value from our customer though is a little different. I'll probably want to do this sort of conversion all over the place, so I have to create yet another method to promote reuse. This time I'll create a method called of_returnValue. that will take a MailReturnCode and convert it to an integer. I like that. I can standardize the return code to the user this way. You'll find the code for this method in Listing 3.

We're finally back to the of_logon method in the server. With the proper setup it's a simple method, just 12 lines (see Listing 4). It finds a hole, converts one of the arguments passed in from CORBA compliant to something the customer component likes, then calls the customer and returns the value returned from the customer (after passing it through our conversion routine).

### Oh, Well...

Again, dear readers, we've run out of space. I'd hoped to get just a little further because our next step ties everything together. Now we have the relationship between the mail server and the mail customer well defined. We're able to connect. In the next article I'll define the relationship between a browser and our mail server. We'll create an environment in which the browser can request a login and get back the headers for all of his mail. From that point on, it's just plain fun. ▼

**Author Bio**

Richard (Rik) Brooks is the owner of Brooks & Young, a Sybase Code Partner. He's been using PowerBuilder since 1990 and has worked as an independent consultant for major corporations in the U.S. for the last five years. He has authored several books on PowerBuilder, including PFC Programmer's Reference Manual and The Definitive DataWindow.
mailReturnCode n_jagMailCustomer.of_logon(as_username, as_password, ao_option)
mailReturnCode imrc_retVal
ib_connected = FALSE
imrc_retVal = ims_mailsession.maillogon(as_username, as_password, ao_option)
if imrc_retVal = mailReturnSuccess! then ib_connected = TRUE
return imrc_retVal

private n_jagMailServer.of_findAHole()
int li_count, li_max
li_max = upperBound(io_customers)
for li_count = 1 to li_max
if not io_customers[li_count].ib_connected then return li_count
next
// No unused customer in the array, create one.
li_max ++
io_customers[li_max] = create n_jagMailCustomer
return li_max

integer n_jagMailServer.of_returnValue(mailReturnCode al_returnCode)
if isNull(al_returnCode) then return -5
choose case al_returnCode
    case mailReturnSuccess!
        return 1
    case mailReturnLoginFailure!
        return -1
    case mailReturnInsufficientMemory!
        return -2
    case mailReturnTooManySessions!
        return -3
    case mailReturnUserAbort!
        return -4
end choose
integer n_jagMailServer.of_logon(as_username, as_password, as_option)
int li_myCustomer
maillogonOption loption
li_myCustomer = of_findAHole()
choose case lower(as_option)
    case "mailnewsession"
        loption = mailNewSession!
        return of_returnValue(io_customers[li_myCustomer].of_logon( as_username, as_password, loption))
    case "maildownload"
        loption = mailDownload!
        return of_returnValue(io_customers[li_myCustomer].of_logon( as_username, as_password, loption))
    case "mailnewsessionwithdownload"
        loption = mailNewSessionWithDownload!
        return of_returnValue(io_customers[li_myCustomer].of_logon( as_username, as_password, loption))
end choose

The employment portal for IT professionals
http://careers.sys-con.com

Dive into your next IT job with confidence!
Mastering the PFC Caching Service

Use the caching service to make your applications run faster

For the past few months this column has focused on the basics of the PFC application/utility services. This month I’ll provide an in-depth look at the most complex of these services: the application caching service.

The application caching service caches data at the application level in DataStores. As a result it can be accessed from window to window. One popular way to use it is to perform a database retrieve from a DataWindow the first time the window opens. Immediately after the retrieve, call a caching function to cache the data in the DataWindow. On subsequent opens of the window you can take the data from the cache, so a database retrieve isn’t necessary.

This service is one of the most powerful in the PFC, so use it often. Rather than retrieving data from a database every time you need it, you can cache it if it’s fairly static. When you need the data it can be taken from the cache without a database retrieve.

Like any application service the developer must first turn it on. In the inherited application manager object open event, type the following code:

This.of_SetdwCache(TRUE)

When the application manager is closed, the caching service is automatically shut off. Developers can do the following with the caching service:

• Cache data
• Retrieve data from the cache
• Inquire about cached objects

Example 1

In this example (see Listing 1) a cached object named customer list is requested from the caching service. If this cached DataStore exists, it’s placed into the requesting DataWindow. If it doesn’t exist, the data is retrieved from the database into the DataWindow, then into the cache. Next time the window opens, the data won’t need to be retrieved from the database. It’ll be taken directly from the cache. Since this DataWindow is updateable, the cached DataStore is updated when the window closes.

Discussion

This window uses Syntax 1 to get the registered DataStore. This syntax allows the developer to access a single DataStore, which is usually declared locally and shared with a DataWindow. This syntax applies here because the cached DataStore and the destination DataWindow have the same column set so a “sharedata” will work. Syntax 1 for of_getregistered() is listed below:

Syntax 1 of_getregistered(): Access a specified DataStore
• Syntax Example:
of_getregistered(ls_id,ln_ds)

• ls_id argument: A string that contains the specified DataStore ID to be returned.
• ln_ds argument: A DataStore passed by reference that will contain the data for the ID specified by ls_id.
• Usage: Use this function when you need to access the data in a single DataStore. You can use the returned DataStore to share data with other DataWindows.
• Return Value: An integer, 1 if OK and -1 if an error occurs.

Before a cached object can be used in the window open event, the function of_isregistered() is called to check if a

Figure 1
specific DataStore exists. The syntax for of_isregistered() is listed below.

Syntax 1 of_isregistered(): Indicates whether a specified ID exists in the cache.
- Syntax Example: of_isregistered(ls_id)
- ls_id argument: A string denoting the ID to be searched for in the cache.
- Usage: Use this function when you want to know if the specified object denoted by ls_id exists in the cache.
- Return Value: An integer, 1 if the ID has been registered, 0 if it hasn’t, and -1 if an error has occurred.

When the window closes it refreshes the cache via the caching services of_refresh() function. This function used to work only with the cached objects that had a retrieve as a data source. Now just about any cached object can be refreshed. The syntax of _refresh() is listed below.

Syntax 1 of_refresh(): Refreshes cached objects.
- Syntax Example: of_refresh(ls_id)
- ls_id argument: A string denoting the ID to be refreshed.
- Usage: Use this function when you want to refresh the data in a cache. If you want to refresh all objects, don’t use an argument (e.g., of_refresh()).
- Return Value: An integer, 1 if the ID has been registered, 0 if it hasn’t, and -1 if an error has occurred.

LOCKING CONSIDERATIONS

Since the DataWindow is linked to the DataStore via the sharedata function, the row and column status of the DataWindow will always be the same as in the DataStore - NotModified. This is the same as when the DataWindow retrieves from the database. If any changes are made to the DataWindow, the user is prompted to save them to the database. The data that will be saved exists in the DataWindow, not in the cache. Normal PowerBuilder rules apply (as far as update characteristics are concerned) (see Figure 1).

How the PFC Refreshes the Cache

After the changes are committed to the database, we must add the code to refresh the cached DataStore. Note The PFC of_refresh() function now works with most cached objects, regardless of how they were cached. Behind the scenes, the caching service is performing a new retrieve into the DataStore. The PFC caching service doesn’t support partial updates in cache via the of_refresh() function. In the example above, if the user changes the last name of the customer, of_refresh() won’t update the row in the DataStore – it’ll perform a full database retrieve. Because of this, before using of_refresh() make sure the database has been updated with any changes. When refreshing the cache, notice that only the cache ID must be passed as an argument. This is because the caching service “remembers” all information about this ID. The service maintains arguments, DataWindow objects, and transaction objects in an object structure. The DataWindow has a sharedata with the cached DataStore in this example, so if we wanted to save any changes the user made in the DataWindow we wouldn’t have to call of_refresh(). Use this scenario when you want the cached DataStore to contain the most recent user changes whether the user saved them to the database or not. But beware, if you do this, the cached DataStore and the database may not always match (see Figure 2).

Example 2

This example is similar to the first (see Listing 2). The only difference is that in this example when of_register() is used, arguments are supplied. Notice that when supplying arguments to the of_register() function, they’re in an array of datatype any. Be sure to specify the correct datatype when assigning the arguments. In Listing 2, a string and a number are passed as arguments to of_register().

Example 3

This example uses syntax 3 of of_register (and caches data that’s held as part of the DataWindow. Of all the of_register() functions, this one is used the least. The reason for this is simple. This of_register() syntax caches data that was kept as part of the DataWindow definition. The data is retained on save only when the DataWindow object was created in the DataWindow painter. This is common when the application is in “prototype” mode and a stable database isn’t available. However, if we’re in a window that contains a DataWindow that contains its own data, we can easily access it without using the caching service; if the window closes and reopens, the data will be displayed anyway. So why cache the data if it’s part of the DataWindow definition? Well, if the data in this DataWindow can be used in other windows, it makes sense to cache it. Again, if you’re using a DataWindow that retained its own data, you’re probably in a demonstration environment and the caching service will be of little help.

Example Code

```
// Cache the data held in the DataWindow
gnv_app.inv_dwcache.of_register("contacts","d_caching34")
```

Window open event:

// Turn on caching if not already on
if Not IsValid(gnv_app.inv_dwcache)
then
  gnv_app.of_setdwcachc(TRUE)
end if

// Cache the data held in the DataWindow

Example 4

This example uses syntax 2 of the of_register() function – it caches a structure (see Listing 3). This can be any structure as long as it matches the columns of the DataWindow object exactly. In the code below, an instance structure is populated with three entries (this is just an example; there are much more creative ways to populate a structure). This structure is then passed to the caching service, which will save it as a DataStore. Remember, once a cached object has been created by the caching service, it’s available throughout the application until the function is called, or the ID is unregistered via the of_unregister() function.

```
string ls_dwobj

If IsValid(gnv_app.inv_dwcache) then
  gnv_app.inv_dwcache.of_register(ls_id, "d_customers")
End If
```

Discussion

As mentioned above, the of_register() syntax is of limited use. The data is already cached, not as part of the caching service but as part of the DataWindow. Since this data is static, it makes little sense to cache it. Database locking is not an issue here since there’s no database (see Figure 3).

Example 5

This example illustrates the other syntax of the of_unregister() functions, as well as caching service functions to unregister cached DataStores (see Listing 4). This example doesn’t use any DataWindows, demonstrating that a DataWindow doesn’t always have to be present.

This code doesn’t contain any functions that haven’t already been covered. However, it checks return values and displays the appropriate messagebox. Also, notice that no DataWindow exists to share data with. Use this code when information needs to be cached in one area of the application for future access. For example, it may be a good idea to cache some sort of static information at the start of an application. While the application is running, this information can be taken from cache, not via another database retrieve.

```
Command Button: cb_unregister clicked
Event:
 // Unregister all objects

int li_rc
li_rc = gnv_app.inv_dwcache.of_unregister()
If li_rc >= 0 then
  messagebox("cache", string(li_rc) + " objects have been unregistered")
Else
  messagebox("cache", "An error has occurred in unregistering cached objects")
End If
```

This code unregisters all DataStores in the cache. By using of_unregister(), a single DataStore can be removed from the service, or if no argument is provided, all DataStores will be removed. This function is commonly used when an ID needs to be refreshed, but of_refresh() can’t be used because the cached ID doesn’t have a retrieval source. In this case, the ID must be unregistered and then registered again. The syntax for of_unregister() is listed below.

Syntax 1 of_unregister(): Remove a specified ID (or all IDs) from cache.
Example 6

This example uses an extended syntax of the of_getregistered() function. This syntax retrieves a DataStore from the caching service and shares the data with a DataWindow of a different column specification. Unlike the other examples, when a DataStore is returned from cache, no sharedata is needed to populate a local DataWindow. The of_getregistered() function automatically transfers the data for you. Use this function when you need to use data from a cached DataStore that has different column specifications. This is valuable when a cached DataStore has many columns, such as the whole employee table. Other DataWindows in the application may not need to share all these columns but may need a subset of them (see Figure 5).

Syntax 4: of_getregistered(): Access a specified DataStore and share it with a DataWindow that has different column specifications.

- Syntax Example: of_getregistered(ls_id_dest_dw)
- ls_id argument: A string that contains the specified DataStore ID to be returned.
- dest_datawindow argument: The destination DataWindow – passed by reference.
- Usage: This use case is similar to using of_getregistered() to access a DataStore from cache. The DataStore returned from cache is treated as a DataWindow. The of_getregistered() function automatically transfers the data for you. Use this function when you need to use data from a cached DataStore that has different column specifications. This is valuable when a cached DataStore has many columns, such as the whole employee table. Other DataWindows in the application may not need to share all these columns but may need a subset of them (see Figure 5).

Example 2: of_getregistered(): Access all cached DataStores.

- Syntax Example: of_getregistered() is listed below:
- ls_list[] argument: An array of strings passed by reference, it will contain all the IDs for the cached DataStores.
- Usage: Use this function when you need to access the names of all the cached DataStores. This function doesn't return the contents of the DataStores – merely their names.
- Return Value: An integer, the number of DataStores, and -1 if an error occurs.

Example 5: of_getregistered(): Access a DataStore and share it with a DataWindow.

- Syntax Example: of_getregistered(dest_datawindow)
- dest_datawindow argument: A DataWindow of a different column specification.
- Usage: This use case is similar to using of_getregistered() to access a DataStore from cache. The DataStore returned from cache is treated as a DataWindow. The of_getregistered() function automatically transfers the data for you. Use this function when you need to use data from a cached DataStore that has different column specifications. This is valuable when a cached DataStore has many columns, such as the whole employee table. Other DataWindows in the application may not need to share all these columns but may need a subset of them (see Figure 5).

Example 1: of_unregister():

- Syntax Example: of_unregister(ls_id)
- ls_id argument: A string denoting the ID to be removed from the cache.
- Usage: Use this function when you want to remove all IDs, don't pass an argument.
- Return Value: An integer, the number of IDs removed. and -1 if an error occurs.
When the user returns to the window, he or she would like to see the most current information that’s been entered. This function takes the contents of the DataWindow and registers them to the cache.

Window open event:
// Get the last customer that was being added
dw_1.insertRow(0)
gnv_app.inv_dwcache.of_register("cust_added",dw_1)

Window close event:
// Cache the contains of the datawindow
gnv_app.inv_dwcache.of_unregister("cust_added")
gnv_app.inv_dwcache.of_register("cust_added",dw_1)

Syntax 4 of of_register() is listed below:

Syntax Example:

Syntax 5 of of_register() is listed below.

Example 8

This example uses syntax 5 of the of_register() function (see Listing 5). This syntax is similar to syntax 4, except that it caches only a row in a DataWindow control, not the whole DataWindow. This extended function is useful when only one of a certain entity can be valid at one time, such as a customer. This example lets the user select an active customer. Once the user clicks a row, the data for that customer is registered to the cache. When the user navigates the application and needs information about this customer, the information can be quickly and easily retrieved from the cache, saving a database retrieve.

Syntax 5 of of_register() is listed below.

Example 7

This example uses an extended syntax of the of_register() function. The PFC doesn’t supply a function that allows the contents of a DataWindow to be cached. Sometimes the most recent information doesn’t reside in the database. When adding a record to the database, what if the data in the DataWindow needs to be cached? For example, a user is entering data in a data entry window and occasionally this window needs to be closed.

Return Value:
An integer, 1 if the ID has been registered, 0 if the ID has already been registered, and -1 if an error has occurred.

Locking Considerations
In this example the DataWindow only interfaces with the database when it adds a new customer. As in the other caching functions, database locking and concurrency is not an issue. However, the status of the row can be an issue. When the user is entering a record, the row has the status of New!. If the user exits the window and the contents of the DataWindow are saved to the cache, when the user reopens the window and the DataWindow is restored the row status will be DataModified!. This will cause a problem when the DataWindow generates the SQL statement to insert a record into the database. Remember, when records are copied from one DataWindow to another, the row status for each respective DataWindow remains the same. To ensure that the row status of the DataWindow is the same as in the DataStore, a sharedata should be used. This example doesn’t use the sharedata since the code to manually change the rowstatus has been added. This was done to illustrate the caution that’s needed when using or extending the caching service.

Final Words

As mentioned before, the caching service is one of the most exhaustive of the PFC. It can be very powerful. It’s always faster to retrieve data from memory than to get it from the database. The use of the caching service will make your applications run faster. Thank you to Kelly Jahneke who works at Disney in Orlando for the topic of this month’s column. I hope this helped.

Bob Hendry is a PowerBuilder instructor at both national and international PowerBuilder conferences. He specializes in PFC development and has written two books on the subject including Programming with the PFC 6.0.
Window open event:
// Create a local DataStore
n_dslds_mydatastore
// Turn on caching if not already on
if Not IsValid(gnv_app.inv_dwcache) then
gnv_app.of_setdwcache(TRUE)
end if
// Create a local DataStore
n_dslds_mydatastore
// Turn on caching if not already on
if Not IsValid(gnv_app.inv_dwcache) then
gnv_app.of_setdwcache(TRUE)
end if
// Get the customer list from cache if it is there. Otherwise, do a retrieve and
// put it into cache for the next time the window opens.
if gnv_app.inv_dwcache.of_isregistered("custlist") = 1 then
gnv_app.inv_dwcache.of_getregistered("custlist",lds_mydatastore)
else
dw_1.Retrieve()
gnv_app.inv_dwcache.of_register("custlist","d_all_emplo-
ees",SQLCA)
gnv_app.inv_dwcache.of_getregistered("custlist",lds_mydata-
store)
end If
lds_mydatastore.ShareData(dw_2)
lds_mydatastore.ShareData(dw_1)
dw_3.Retrieve()
Window close event:
// Refresh the customer list. Get the DB version
gnv_app.inv_dwcache.of_refresh("custlist")

Window open event:
// local variables
anyla_args[]
n_dslds_mydatastore
// Arguments to pass to the caching service
la_args[1] = 501
la_args[2] = "F"
// Turn on caching if not already on
if Not IsValid(gnv_app.inv_dwcache) then
gnv_app.of_setdwcache(TRUE)
end if
// Get a list of women who work for manager #501 if it is there. Otherwise, do a retrieve
// and put it into cache for the next time the window opens.
if gnv_app.inv_dwcache.of_isregistered("emplist501") = 1 then
gnv_app.inv_dwcache.of_getregistered("emplist501",lds_mydata-
store)
else
dw_1.Retrieve(501,"F")
gnv_app.inv_dwcache.of_register("emplist501","d_employees",SQLCA,la_args[])
gnv_app.inv_dwcache.of_getregistered("emplist501",lds_mydata-
store)
end If
lds_mydatastore.ShareData(dw_1)
Command Button: cb_register clicked event:
// local variables
intli_rc
// Cache the customer list
li_rc = gnv_app.inv_dwcache.of_register("ex5custlist","d_custom-
ers",SQLCA)
if li_rc = 1 then
messagebox(“Cache”,“The customer list has been cached”)
else
if li_rc = 0 then
messagebox(“Cache”,“The customer list cannot be cached.
It already exists”)
else
messagebox(“Cache”,“An error has occurred trying to
cache the customer list”)
end If
End If
// cache the employee list
li_rc = gnv_app.inv_dwcache.of_register("ex5employees","d_all_emplo-
ees",SQLCA)
if li_rc = 1 then
messagebox(“Cache”,“The employee list has been cached”)
else
if li_rc = 0 then
messagebox(“Cache”,“The employee list cannot be cached.
It already exists”)
else
messagebox(“Cache”,“An error has occurred trying to
cache the employee list”)
end If
End If
DataWindow (dw_1) clicked event:
// unregister the current cust if it exists
gnv_app.inv_dwcache.of_unregister("current cust")
// Cache only one row in the DataWindow
gnv_app.inv_dwcache.of_register("current cust",This,row)
// Get the cached row from the DataWindow and transfer in to
// DataWindows of different column specifications
gnv_app.inv_dwcache.of_getregistered("current cust",dw_2)
gnv_app.inv_dwcache.of_getregistered("current cust",dw_3)
The Many Lives of PowerBuilder Developers

Career opportunities within the expanding Web enterprise

WRITTEN BY SUE DUNNELL

There’s no denying it. We’re living in an n-tier world, especially since the release of PowerBuilder 7.0 and the Web DataWindow. It’s almost counterintuitively caused many PB developers to ask themselves questions like: “What am I going to do with my client/server PowerBuilder skills? Where is the market going? Do I have to learn Java and JavaScript? Or C++? Or VB and VBScript?”

As the manager at Sybase most immediately responsible for the future of PowerBuilder, I’m going to offer some clear-cut answers to these questions.

Future Lives, Future Livings: Careers for PowerBuilder Developers

There are three distinct, yet interrelated, career paths available to PowerBuilder developers:

1. Creating and maintaining client/server PowerBuilder applications
2. Migrating your PowerBuilder code to the Web, and...
3. Don’t look now – you’re a Web developer!

With these three rich career paths in mind, ask yourself now which kind of PowerBuilder developer you’d like to become. How far would you like PowerBuilder to take you?

Client/Server: Like a Rock

Client/server is very much alive; organizations around the world have amassed mountains of client/server code. Because the Web paradigm is inappropriate for many in-house projects (these applications generally don’t demand the scalability associated with e-commerce and middleware), a significant number of applications will continue to be built according to the stable, proven, client/server model.

POWERBUILDER 8.0: MORE FUNCTIONALITY, PRODUCTIVITY FOR CLIENT/SERVER

Anticipating your ever-increasing need for speed, we’ve built a number of important new or enhanced productivity features into the next major release of PowerBuilder. Chief among these:

- **Library painter:** Now also displays all the objects in your computer’s file system
- **Improved exception handling:** Similar to the implementation found in Java
- **Build and deploy PowerBuilder applications from the command line:** Schedule compiles
- **Autoscript**
  - **Source code control:** Improved team development
  - **Powerscript:** New file and printer system functions

If your working life revolves around client/server development, rest easy. Armed with your current PowerBuilder skills, you’re well prepared to meet the development challenges of the future.

You Can Take It with You: PowerBuilder Code in Your Next Life

Are you thinking about migrating your client/server applications to the Web? Good news: you don’t have to start from scratch, incurring the risk and expense you’d expect. Happily, you can take your existing PowerBuilder encapsulated objects with you and deploy them to Sybase EAServer as PowerBuilder components.

“LEGACY” POWERBUILDER CODE

Perhaps you’re concerned about the health of your code. Maybe you’re working with an undocumented PowerBuilder 6 application — and the original developer retired to Costa Rica. Depending on the scope of your endeavor, we’ve designed solutions to help you get from tiers 1 and 2 to tier n:

- **Online Documentation:** Most migration issues are readily addressed by Sybase reference materials, available online at [www.sybase.com/support](http://www.sybase.com/support).
- **Sybase Professional Services (SPS):** My colleague, practice manager Mitch Federman, recently explained to me that his professional services team was able to show a defense-industry client how Sybase’s eMAP migration services “would reduce their time to deployment, trim their development costs, and eliminate the risk associated with rewriting their PowerBuilder code.” The client had previously consulted other application server vendors, all of whom said they’d have to recode in Java. Instead, SPS showed the
firm how to “leverage its PowerBuilder investment by identifying which PowerBuilder applications and objects to migrate to the [EAServer] application server” and which to leave alone. (SPS is available for customized consulting and turnkey services.)

"NEW" POWERBUILDER CODE COEXISTING WITH EJB, CORBA, COM

On the other hand, if your business logic is already stored in non-visual objects, you’re ready to begin a new life as a developer of multtier applications.

PowerBuilder 7 and the upcoming version 8.0 dramatically simplify the process of migrating to the middle tier. Easy-to-use wizards allow you to deploy your modified PowerBuilder NVOs to Sybase EAServer (where they can coexist with a variety of other component types) – making your previously coded business logic available to a host of new Web, Java, C++, and other applications via the application server.

To make your life even easier, we’ve included a development edition of Sybase EAServer in the box with PowerBuilder 8.0.

The Promised Land - Here Already?

You’re a Web developer and you didn’t even know it

With PowerBuilder, the transition from client/server developer to Web developer is no great leap. Everything you need is in the box: Web DataWindow, application server, HTML generator, migration wizards, and seamlessly integrated PowerSite functionality (v8.0).

A SINGLE, UNIFIED DEVELOPMENT ENVIRONMENT

Now, with PowerBuilder 8.0, you can develop and deploy business logic to EAServer with 4GL ease. You can create an HTML client interface (complete with autoformatting, client-side validation, calculated fields, and managed database connections) without leaving PowerBuilder.

The Big Opportunity

PowerBuilder’s new functionality is exciting because it allows your PB code to execute alongside Java, C, C++, and other components within the EAServer application server platform. This puts you in direct proximity to new career opportunities within the expanding Web enterprise.

All because you develop with PowerBuilder. ▼

Author Bio

Sue Dunnell is the product manager for PowerBuilder at Sybase. Previously, she spent three years in the PowerBuilder Custom/Alliance Support group. Later she held a staff position in Support dedicated to internal training, communication, hiring, and customer service. Sue works in Concord, Massachusetts.

dunnell@sybase.com
As more PowerBuilder applications are being converted to Java, I wrote this article to provide insight to those who might be involved in such a project. My experience comes from one particular project where the goal was to convert an order entry system written in PowerBuilder 6.5 to a Java 2 (1.2) application using Swing components.

This article is for an experienced PowerBuilder developer with an intermediate knowledge of the Java language and object-oriented programming concepts. It provides insights only into the issues that may be involved in converting a PowerBuilder application to Java. Java is a different technology and therefore has many different issues and concerns that need to be dealt with. This article shouldn't be used as a replacement for proper object-oriented design and system analysis. Also, any Java project should include one or more experienced Java developers who have object-oriented design and implementation experience.

Parts of the conversion were straightforward since PowerBuilder supports many object-oriented capabilities such as inheritance, polymorphism, and encapsulation. These features translate nicely into Java. Other parts weren't so easy, as PowerBuilder is a fourth-generation language with many high-level functions built into it. These features aren't so easy to implement in Java, which is considered by most to be a third-generation language. When I use the term Java, I'm referring to the core Java language along with the standard Java application programming interfaces (APIs). Java, along with the standard APIs, is also referred to as the standard Java edition. Other editions of Java include other APIs, but they're not discussed here.

When converting PowerBuilder to Java, it's beneficial to have a good understanding of the differences and similarities between the two technologies. In the next section I'll briefly describe the project I was
involved in. In the following two sections we'll take a look at the mappings of datatypes and controls from PowerBuilder to Java. We'll then look at conversion topics and some examples of how PowerBuilder translates into Java. At the end we’ll cover ideas on how a PowerBuilder developer should prepare for a Java project.

### Project Description

The order entry system is a tab-based, single-document interface that utilizes non-SQL DataWindows, command buttons, and response windows. The DataWindows contain single-line edits, DropDownDataWindows, lists, check boxes, radio buttons, and group boxes. The application retrieves and saves its data through a C++ Windows DLL library of functions that communicates with a UNIX server running an Oracle database.

My client had two main goals in converting the PowerBuilder application to Java. The first was to allow for easy access to the application over the Internet. To achieve this, we implemented the application as an applet that required users to install the Java plug-in. The second was to utilize standard Web technologies. My client perceived Java as a standard Web technology they could load onto their users' systems. The new Java application retrieves and saves its data through IBM’s WebSphere application server. This application server won’t be discussed here since any other compatible server could have been used instead.

The Java integrated development environment (IDE) used for this project was IBM’s VisualAge for Java 1.2 version 3.0. Since Java is an open language supported by a large number of software vendors, the IDE isn't critical. I don't focus on the IDE since any one could be used for a conversion project. If you're interested in IDEs, there are a number to choose from. Here's a list of a few of the most popular:

- Borland JBuilder
- IBM VisualAge
- Oracle JDeveloper
- Sun Forté
- Sybase PowerJ
- WebGain VisualCafé (formerly Symantec VisualCafé)

### Datatype Mappings

The conversion of the basic datatypes from PowerBuilder to Java is fairly easy (see Table 1). Note: The integer datatype in PowerBuilder converts to the short datatype in Java, and the long datatype in PowerBuilder converts to the integer datatype in Java. To keep things simple in our project we converted both the PowerBuilder integer and long datatypes to the integer datatype in Java. The conversion of real and decimal datatypes can be tricky if precision is an issue. Research and test the datatypes in both PowerBuilder and Java to be sure you're getting the results you want.

### Control Mappings

The conversion of graphical controls from PowerBuilder to Java is not quite as easy. Picking a control in Java to replace the PowerBuilder control can usually be done without too much effort. However, for many controls you may have to be creative. Sometimes there can be more than one choice. The tricky part usually has to do with the conversion of the events associated with the control. In our project we soon learned not to get hung up on making controls work exactly the same between PowerBuilder and Java, but instead making the Java controls achieve the same business purpose as their PowerBuilder counterparts.

Table 2 provides suggested mappings of PowerBuilder controls to Java; don't be afraid to stray from this list. Many possible mappings are available. You may even disagree with some of them. Also, I selected Swing components over AWT components whenever possible.

### Conversion Topics

This section covers various topics and issues you may encounter when converting PowerBuilder to Java. Note: These examples reflect the way we did things on our project. As with any programming assignment, there are always many ways to accomplish a task.

#### JAVA VERSIONS

One of the first things you need to decide for your project is what version of Java to use as there are many different versions available. The most current version of Java is Java 2 version 1.3. The term Java 2 may be confusing to you. Between the 1.1 and 1.2 versions of Java, Sun changed the name of Java to Java 2. You may need to use other Java software products, such as the Java plug-in, and make sure any other Java software you use is compatible with your version.

#### LOOK AND FEEL

PowerBuilder developers usually write applications that run on the Windows operating system. However, Java applications are often meant to run on many different systems including Windows, UNIX, Linux, and Macintosh. Because of this, the creators of Java created a concept called Pluggable Look and Feel. Programmers have the ability to change the way their applications look and feel (behave) at runtime. The default look and feel for Java is the Java (Metal), while others exist that look like Windows, Motif, and Macintosh. The Java or Metal look and feel is similar but distinctly different from Windows'.

PowerBuilder developers tend to want to make their Java applications look like Windows applications. Two ways to achieve this are (1) use the Windows look and feel and (2) modify the User Interface Manager (javax.swing.UIManager) object. The problem with the first solution is the Windows look and feel is only available for Windows platforms because of trademark reasons. The problem with the second solution is the developer must make manual changes to the look and feel and will be left with a mixed-up look and feel somewhere in between Java and Windows. Also, the Java look and feel is recommended by Sun for cross-platform development. Sun claims the Java look and feel will be the most stable across the various operating systems that Java runs on.

For our project we decided to go with the Java look and feel. We planned for our application to run on more than one operating system and didn't want to be bothered with the details of manually programming the look and feel for each individual component. If you decide to use the Java one for your application, you may want to let your users know of your plans up front so they don't expect the Windows look and feel when they deliver the finished product.

<table>
<thead>
<tr>
<th>DATATYPE</th>
<th>POWERBUILDER</th>
<th>JAVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>Character</td>
<td>char or character</td>
<td>char</td>
</tr>
<tr>
<td>Date</td>
<td>date</td>
<td>java.util.Date</td>
</tr>
<tr>
<td>Date Time</td>
<td>datetime</td>
<td>java.util.Date</td>
</tr>
<tr>
<td>Decimal</td>
<td>dec or decimal</td>
<td>float/double (if precision is a concern, be sure to thoroughly research and test these data types)</td>
</tr>
<tr>
<td>Integer</td>
<td>int or integer</td>
<td>short</td>
</tr>
<tr>
<td>Long</td>
<td>long</td>
<td>int</td>
</tr>
<tr>
<td>Real</td>
<td>real</td>
<td>float/double (if precision is a concern, be sure to thoroughly research and test these data types)</td>
</tr>
<tr>
<td>String</td>
<td>string</td>
<td>java.lang.String or java.lang.StringBuffer</td>
</tr>
<tr>
<td>Time</td>
<td>time</td>
<td>java.util.Date</td>
</tr>
</tbody>
</table>

TABLE 1: Mappings of basic datatypes from PowerBuilder to Java
LEARN ABOUT SWING COMPONENTS

Swing components are significantly different from PowerBuilder components. One of the most significant differences is that most of them are built using the model-view-controller architecture (MVC), which allows the data and presentation layers to be separated during development and for more flexible design. Learning how Swing and MVC work together will greatly enhance your Java programming abilities.

Other differences between Swing and PowerBuilder components are subtle and may lead to confusion. One example is the setEnabled method used by JTextField in Java. PowerBuilder doesn't use that method; it uses setEnabled for disabling a component. JTextField also uses a setEnabled method in addition to setEditable. Calling setEnabled(false) on a JTextField will prevent the user from entering new text in the component. However, setEditable(false) will need to be called in order to make the background of the JTextField gray.

LAYOUT MANAGERS

Layout managers are an important part of the Java graphical user interface (GUI) development environment. They're complex and all Java GUI developers should be familiar with them. I won't go into the details of layout managers. The main purpose of addressing this topic here is to encourage you to use a layout manager.

If the layout manager of a Java container is set to null, then the container will honor all x and y coordinates and widths and heights for various components set by the programmer. If you don't use a layout manager, the application may not display properly on a different operating system or even run properly. Even though layout managers take a while to learn, they're well worth the effort. The only time a null layout manager is acceptable is when you're creating a prototype and have very limited development time.

Not only do layout managers help applications adapt to different operating systems, they also relieve the programmer from having to perform the tedious task of setting x and y coordinates along with the widths and heights for graphical components.

For most good IDEs, the grid bag layout is the best choice for layout manager. It's one of the most complicated of the layout managers, but also one of the most flexible and powerful. Most good IDEs generate the code needed for using grid bag layout.

POWERBUILDER UNITS TO PIXELS

If you're converting visual components from PowerBuilder to Java, you may want to keep some of the same look and feel. This may include making your new Java components a size similar to what they were in PowerBuilder.

One way of doing this is by converting the PowerBuilder units (PowerBuilder component measurements) to pixels (Java measurements). PowerBuilder provides two nice functions for doing this:

- XUnitsToPixels
- YUnitsToPixels

We actually used a small PowerBuilder calculator application that utilizes these two functions to convert PowerBuilder units to pixels.

REFERENCE PARAMETERS

Java doesn't support reference parameters. Parameters are passed by value in Java. To simulate passing a parameter by reference, you must pass an object as an argument and then update data inside the object.

Listing 1 provides a class definition in Java for the TeamInfo object. The following code calls the lookupTeamInfo method, gets data out of the teamInfo variable, then prints it.

```java
String memberName = "John Smith";
TeamInfo teamInfo = new TeamInfo();
LookupTeamInfo (memberName, teamInfo);
System.out.println(teamInfo.getTeamName());
System.out.println(teamInfo.getTeamLeaderName());
System.out.println(teamInfo.getNumberOfMembers());
```

Warning: Strings and the standard wrapper classes don't allow their values to be updated (this is called immutable). Therefore they can't be used for passing data in this way. Instead, make your own wrapper classes or use arrays of objects. For strings you can use the StringBuffer instead.

The wrapper classes wrap the primitive datatypes int, long, char, and more. For example, the Integer class wraps the int datatype. Once the value for this class is set, it can't be modified. Instead, create a custom class such as ReferenceInteger that you can use for passing integer values by reference.
FROM STRUCTURES TO OBJECTS

PowerBuilder has a structure datatype. Java doesn’t. A common way of creating structures in Java is to create a class and add instance variables for each field in the structure. These fields should be declared private. To access them, getter and setter methods should be created. The TeamInfo class in the Reference Parameters example is a good example of a Java structure that has getter and setter methods.

PASSING DATA TO AND FROM WINDOWS

In PowerBuilder, you passed parameters to dialog (response) windows in the Open function, and retrieved data from dialog windows using the Message object. In Java, passing data to and from dialog windows is a little different; however, I think you’ll like the change.

To create and open a dialog window in Java, first create the dialog and then make it visible. When a dialog becomes visible, it pops up on the screen for the user to see and interact with. You must first set the dialog to modal to force the user to interact only with the dialog and not the rest of the application.

To pass data to the dialog prior to making the dialog visible, call setter methods on the dialog object. The setter methods are added to the dialog class in the same manner as in the Reference Parameters section. When the dialog becomes visible, the windowOpened method is triggered (see java.awt.WindowListener for more information on window events). In the windowOpened method you can access the variables that were previously set in your code prior to making the dialog visible.

To pass data back from the dialog to the calling script, the dialog sets values to its variables and then makes itself invisible. When a dialog is no longer visible, the modal effect ends and processes transfers back to the calling script. The calling script can then call getter methods on the dialog to get the data out of it.

To discard the dialog, set its reference to null and garbage collection will clear the memory allocated for it. If you don’t set the reference to null, when the script finishes processing, garbage collection will clear the memory for the dialog as long as there are no other references to it. Actually, you can never be sure when garbage collection will clean up unused objects. Garbage collection is another subject entirely and is explained in most introductory Java books.

Here’s an example of passing data to and from a dialog.

```java
myDialog = new JDialog();
myDialog.setModal(true);
myDialog.setLastName("Gosling");
myDialog.setFirstName("James");
myDialog.setModal(true);
JDialog myDialog = new JDialog();
// Create the dialog and make it modal.
myDialog = new JDialog();
myDialog.setModal(true);
// Pass data to the dialog.
myDialog.setFirstName("James");
myDialog.setLastName("Gosling");
// Display the dialog to the user.
myDialog.setVisible(true);
// Get the return value from the dialog.
rtnCode = myDialog.getReturnCode();
// Get rid of the dialog.
myDialog = null;
```

CONVERTING DATAWINDOWS

Of all the PowerBuilder objects, DataWindows can be the most complicated to convert to Java. They’re the most used and functional of all the PowerBuilder components. However, they’re probably the most unorthodox. It’s well worth your time to analyze how your DataWindows have been used and what functionality they provide, then determine a plan for converting their functionality to Java. There are many different uses of DataWindows and too many to list here; I’ll address only two of them.

• Form DataWindows: The most straightforward way is to create a JPanel and add Java controls to it that are similar to the PowerBuilder controls in the DataWindow. (See Table 2 for a list of control mappings between PowerBuilder and Java.) Remember, a number of DataWindow attributes and expressions can be set on controls inside DataWindows.
• List DataWindows: The JTable is probably the best choice for most list DataWindows. However, you may choose to use other controls such as swing.JList to implement a single-column list in a DataWindow. You may even decide that a JTree can be used to replace some of them. The choice is up to you.

CONVERTING A DROPDOWNLISTBOX

The DataWindow’s DropDownListBox in PowerBuilder supports both display and data values. The latter is used for associating the former to a unique code used for storage (usually in a relational database). For DropdownLists Java has the JComboBox. JComboBox doesn’t support data values, only the display value. To overcome this deficiency, you’ll need to extend JComboBox and add your own support for data values.

ADDING A DROPDOWNLISTBOX TO A JTABLE

You may be tempted to convert many of your DataWindows to JTables. JTables are easy to set up for basic functionality. However, be careful to set aside enough time to work on advanced JTables. Adding advanced features to JTables usually involves some extra research and experience. Here’s an example of how to add a JComboBox to a JTable. You’ll need this when you want to display a list of data and a DropDownList in one of the columns.

```java
import javax.swing.JTable;
import java.awt.

public class SampleTable extends JTable {
   // Constructor
   public SampleTable() {
      // Get third column in table.
      javax.swing.JTable column = this.getColumnModel().getColumn(2);
      // Set third column in table.
      JComboBox comboBox = new JComboBox();
      DefaultCellEditor cellEditor = new DefaultCellEditor(comboBox);
      column.setCellEditor(cellEditor);
      DefaultCellEditor cellEditor = new DefaultCellEditor(comboBox);
      column.setCellEditor(cellEditor);
      this.setTableHeader(column, cellEditor);
      myPanel.setBorder(new javax.swing.border.TitledBorder("My Panel"));
   }
   // ButtonClicked Event:
   for (Event e : this.getEvents()) {
      if (e instanceof ButtonPressedEvent) {
         ButtonPressedEvent event = (ButtonPressedEvent) e;
         System.out.println("Button Pressed: "+ event.getMessage());
      }
   }
}
```

CONVERTING EVENTS

Events are one of the most complicated issues when converting PowerBuilder to Java. There are many ways to map events from PowerBuilder to Java. There are also a large number of events to learn to understand what events are even possible in Java. Here are three examples of events we converted from PowerBuilder to Java.

• ButtonClicked Event: For this event we used the ActionListener.actionPerformed event.
• Window Opened Event: For this event we used the WindowListener.windowOpened event.
• DataWindow ItemChanged Event: For this event we used the FocusListener.focusLost event. We converted most of the form style DataWindows to JPanels with JTextField on them. JTextField is a descendant of java.awt.TextField, which is capable of trapping the FocusGained and FocusLost events. The
A common way of creating structures in Java is to create a class and add instance variables for each field in the structure.

COMPARING STRING VALUES

A common pitfall for an inexperienced Java programmer is to incorrectly compare String variables using the “==” comparison operator instead of the “equals” method. Strings are objects and when “==” is used to compare two objects, the expression checks if the objects are the same. The expression doesn’t check that the contents of both objects are the same. All classes in Java inherit from the Object class. The Object class has a method, “equals,” which can be used for evaluating the contents of the object. The String object overrides this method for that exact purpose.

```java
if (employeeName.equals(managerName)) {
    // Correct
}
```

Note: You can use the “==” comparison operator under special circumstances to compare strings for equality. However, this is an advanced technique that should be used only by experienced Java programmers.

DATE AND TIME FUNCTIONS

PowerBuilder offers some nice date and time functions as part of the core PowerScript language and the PowerBuilder Foundation Classes (PFC). In addition, PowerBuilder has both the date and time datatypes.

Instead, Java has the Calendar and Date components in the java.util.package. These classes are full featured and can deliver the same functionality achieved in PowerBuilder. You’ll want to study both of these classes to determine how they function. For our project we built functions that imitated the PowerBuilder functions on top of these classes.

DO NOT IMITATE TOO MUCH

In the previous section about Date and Time functions, I mentioned that we imitated some of the PowerBuilder date and time function in Java. While this can be beneficial for some of the more complex functions you’re used to using, it can be a bad thing to do when trying to learn Java. We made a conscious effort not to imitate too many of the PowerBuilder functions because we wanted our PowerBuilder developers to learn the Java way.

An example of a PowerBuilder function you may be tempted to imitate in Java is MessageBox. We decided not to imitate it because we wanted everyone to learn how to use JoptionPane in Java.

CONVERTING THE MESSAGEBOX

Since we’re on the topic of MessageBoxes, the conversion of a MessageBox from PowerBuilder to Java is simple. Here’s an example.

```java
public interface Constants {
    public final static int FAILURE = -1;
    public final static int SUCCESS = 1;
}
```

```java
public class MyClass implements Constants {
    public final static int SUCCESS = 1;
    public final static int FAILURE = -1;
}
```

```java
public class MyClass implements Constants {
    public final static int SUCCESS = 1;
    public final static int FAILURE = -1;
}
```

```
if (myMethod() = Constants.FAILURE) {
    return;
}
```

The second is not as well known but has a nice feature. Instead of using a class with final static member variables, use an interface. By definition, Java makes all interface variables final and static. If your class extends a class or an interface, when you reference a member variable you don’t have to specify the name of the class or interface before the name of the variable. Also, if there are ambiguities between variable names, the Java compiler will force you to eliminate them. You don’t want to extend another class (as in the first example) simply for this feature. However, classes can implement as many interfaces as you want. The one drawback of using an interface is that the variables are always final. Here’s the same example as above using an interface instead.

```java
public interface Constants {
    public final static int FAILURE = -1;
    public final static int SUCCESS = 1;
}
```

```java
public class MyClass implements Constants {
    public final static int SUCCESS = 1;
    public final static int FAILURE = -1;
}
```

```java
if (myMethod() == Constants.FAILURE) {
    return;
}
```

IMPORTANT CUSTOM COMPONENTS

Some components that you’d want to use in your Swing applications will take considerable time to develop. You can try to develop these components yourself or save time by purchasing them from third-party software vendors. The prices are usually affordable, and many of the vendors also sell the source code. Two of the components we acquired from third-party vendors include both a calendar and an edit mask object. Many components are also available for free.

Some Web sites to help locate third-party components are:
- www.componentsource.com
- www.flashline.com
- www.freewarejava.com
- www.gamelan.com
- www.jars.com

DOCUMENTING YOUR CODE

JavaDoc is Java’s self-documentation feature built into Java. First, program JavaDoc-compatible comments right into your code. Then,
when you're ready, generate HTML documentation from these comments using the JavaDoc utility that comes with the Java SDK. Therefore, when you finish coding your application, much of your documentation will already be completed.

PRINTING

This topic probably caused us the most problems on our project. PowerBuilder does a great job at printing. In particular, the DataWindow is excellent for printing and creating reports. However, printing is probably one of the weakest areas in Java. Java also wasn't designed to be a reporting tool. Therefore, converting PowerBuilder reports to Java is a very difficult task.

To assist you in this area, you'll want to find printing examples on Sun's Java Web site and other sites. You'll also want to consider purchasing third-party Java printing components and maybe even a Java reporting tool.

Because printing in Java is such an issue, look into your printing requirements early so you can plan what approach to take.

Preparing Yourself

This section covers various topics that can help you prepare for a PowerBuilder-to-Java project.

JAVA, SUN.COM

You should become very familiar with Sun's Java Web site. They have one of the best Web sites designed for developers that I've found. You'll find programming tips, tutorials, resources, bug lists, and news about Java and its upcoming releases. Their site also has links to other sites dedicated to Java.

JAVADOC

The standard Java HTML documentation was itself generated using JavaDoc. All team members should be familiar with JavaDoc and its contents. It's an excellent resource and can increase your productivity. JavaDoc for each version of Java is available on Sun's Java Web site along with the Java Development Kit (JDK).

RESOURCES

To keep up with the new Internet technologies that are continually changing, you need to become an avid reader. There are many excellent books that are fit for beginning developers all the way up to Java experts.

Here are a few to get you started:

• Eckel, B. Thinking in Java
• Horstmann, C., and Cornell, G. Core Java 2, Volume 1
• Horstmann, C., and Cornell, G. Core Java 2, Volume 2: Advanced Features
• Flanagan, D. Java in a Nutshell: A Desktop Quick Reference
• Eckstein, R., Loy, M., and Wood, D. Java Swing
• Chan, P., and Lee, R. The Java Developer's Almanac 2000

SUN'S JAVA CERTIFICATION

Becoming Java certified not only helps build your résumé, but will also most likely improve your Java skills. Unless you have a tremendous amount of Java knowledge already, studying for the certification exam will strengthen your existing skills and teach you a few things you didn't already know.

Certification exams can be intimidating to some. Even if you don't plan on taking the test, reading a certification study guide can be a worthwhile exercise. The guides cover the basic Java language features you'll use on all Java projects.

Here are a few of the popular Java certification study guides:

• Brogden, B. Java 2 Exam Prep
• Brogden, B. Java 2 Exam Cram

Information about taking the official Sun Java certification test can be found on Sun's Java Web site.

Conclusion

Every project needs a good plan. Hopefully, we've covered some ideas that can help you create that plan. I believe the most important ideas covered in this paper are getting your PowerBuilder developers trained in Java, adding experienced Java developers to your project, and understanding the differences and similarities between PowerBuilder and Java.

Author Bio

Robert Breidecker, a senior consultant for Ferguson Consulting in St. Louis, Missouri, has over 10 years of experience working in the IT industry. A Powersoft certified PowerBuilder associate, he's also a Sun certified Java programmer for the Java 2 Platform.

Listing 1

```java
public class TeamInfo {
    private String teamName;
    private String teamLeaderName;
    private int numberOfMembers;

    public String getTeamName() {
        return teamName;
    }

    public String getTeamLeaderName() {
        return teamLeaderName;
    }

    public int getNumberOfMembers() {
        return numberOfMembers;
    }

    public void setTeamName(String newTeamName) {
        teamName = newTeamName;
    }

    public void setTeamLeaderName(String newTeamLeaderName) {
        teamLeaderName = newTeamLeaderName;
    }

    public void setNumberOfMembers(String newNumberOfMembers) {
        numberOfMembers = newNumberOfMembers;
    }

    public void lookupTeamInfo(String memberName, TeamInfo teamInfo) {
        if (memberName.equals("Mary Smith")) {
            teamInfo.setTeamLeaderName("Finance");
            teamInfo.setTeamName("Mary Jones");
            teamInfo.setNumberOfMembers(12);
        } else {
            teamInfo.setTeamName("Unknown");
            teamInfo.setTeamLeaderName("Unknown");
            teamInfo.setNumberOfMembers(0);
        }
    }
```

The code listing for this article can also be located at www.PowerBuilderJournal.com
Dear Cliff,

You would salivate at the stuff I’m doing. If you thought PowerBuilder was nice, Jaguar blows it clean away. The stuff I’m doing with Jaguar is beyond belief.

Just to catch you up, Jaguar is an application server. Actually it’s only one small part of a piece of software that comes bundled from Sybase called Enterprise Application Server. Along with it comes a Web server with its own scripting language (PowerDynamo) and a bunch of tools for translating between one protocol and another.

Essentially you create components. These are just nonvisual objects. You can write them in PowerBuilder, Java, C/C++, or any language that generates CORBA objects. In fact, since EAServer is J2EE compliant, you can even host EJBs on it. Then, using something called the Jaguar Manager, you deploy the objects to Jaguar. Of course, if you’re using a Sybase development tool, you can do it from that.

Once it’s on Jaguar, you can access it in several different ways. You have to know the TCP/IP address of Jaguar, though. It’s easy to do it from a Sybase development tool. Click on New, then select a wizard to generate local proxies. This gives you an object you can use in your application like any other NVO.

Before you can use the proxy, you have to connect to Jaguar, where another wizard steps you through generating a connection object. It, too, winds up looking like a proxy. Here’s the code to use that connection object:

```
   n_myConnection_object = go_jaguar() // assuming you are defining it as global
```

Then in your script put the following lines:

```
   go_jaguar = create n_myConnection_object
   if go_jaguar.of_connectToServer() = -1 then
     messagebox("ERROR", "Could not connect")
     halt close
   end if
```

To use a proxy, you define it as a variable (just like any other NVO) and create it, but since it’s CORBA you can’t just use the CREATE. Here’s what you do:

```
   n_jaguarObject = local Object
   go_jaguar.createInstance(local Object)
```

Now you can use your proxy as if it was a PowerBuilder NVO.

It’s sort of like the paging we used to do years ago when writing word processors in C. Remember that? You’d read a part of the file that was edited and try to keep your current index somewhere in the middle of that to minimize the “hicups” the user would see when doing page down and up. It’s pretty much the same concept.

The shared objects don’t go out of scope, and there’s never more than one of these at a time in Jaguar memory, since that one is shared. It fires off a separate thread for every process that calls it.

The services objects allow you to create something akin to Windows services or UNIX CRONs, except that it runs entirely within the context of Jaguar. It fires off and just keeps running forever, sleeping for a programmer-defined period after each run. There’s a function call to make it sleep (JagSleep).

So these are the types of components. The story doesn’t end here though. Inheritance is fully supported so you have all the power of inheritance in your components. Mine, for example, I created a base class and then added code that would write to the server log file. Of course, that wasn’t quite enough (when is anything enough?). I read an article in PowerBuilder Developer’s Journal in which Mike Barlotta described how to use Jaguar component properties. In this way, whoever is charged with maintaining Jaguar can fire up an application called the Jaguar Manager, find a package and a component within that, then right-click on it and add a new property. The property, like any other, has a name and a value except that it’s a string. From within my component I can get that property with a function called getContextKeywords. It returns me to an array of strings that are the values for all the properties the Administrator set up by a particular property name in the Jaguar Manager. I try to get the Administrator never to set up more than one, but he or she may set up a property named “foo” on more than one component, in which case I get one array element for each of them.

I have a ServicesManager object that’s really just a shared object, not a service at all. Basically, the idea is that the programmer can register (taking a page from PFC) objects that he or she inherits from a component I created with this manager. I’ll store the objects in an array and check them periodically. If, from the internal properties of the component, it finds one that’s ready to fire, it fires the component run method and resets the next time it’s due to run.

I’ve got this set up so that each runnable component can be set up to run on specific days of the week, days of the month, or every day. The programmer can also specify a start and end time along with an interval at which to run the component run method.

Author’s comment: This is a message that I sent to Cliff High, an old friend and fellow programmer. We’ve worked together from time to time for the last two decades – mostly with C++ contracts – but recently with a PowerBuilder contract. I helped him learn PowerBuilder through e-mail and he was strongly impressed with the language, especially the ability to create truly object-oriented applications. This contract wasn’t EAServer so he wasn’t exposed to it. The message above is in response to his asking me how I like my current EAServer contract.

---

I M H O

Letter to a Friend

WRITTEN BY RICHARD BROOKS

Dear Cliff,

You would salivate at the stuff I’m doing. If you thought PowerBuilder was nice, Jaguar blows it clean away. The stuff I’m doing with Jaguar is beyond belief.

Just to catch you up, Jaguar is an application server. Actually it’s only one small part of a piece of software that comes bundled from Sybase called Enterprise Application Server. Along with it comes a Web server with its own scripting language (PowerDynamo) and a bunch of tools for translating between one protocol and another.

Essentially you create components. These are just nonvisual objects. You can write them in PowerBuilder, Java, C/C++, or any language that generates CORBA objects. In fact, since EAServer is J2EE compliant, you can even host EJBs on it. Then, using something called the Jaguar Manager, you deploy the objects to Jaguar. Of course, if you’re using a Sybase development tool, you can do it from that.

Once it’s on Jaguar, you can access it in several different ways. You have to know the TCP/IP address of Jaguar, though. It’s easy to do it from a Sybase development tool. Click on New, then select a wizard to generate local proxies. This gives you an object you can use in your application like any other NVO.

Before you can use the proxy, you have to connect to Jaguar, where another wizard steps you through generating a connection object. It, too, winds up looking like a proxy. Here’s the code to use that connection object:

```
   n_myConnection_object = go_jaguar() // assuming you are defining it as global
```

Then in your script put the following lines:

```
   go_jaguar = create n_myConnection_object
   if go_jaguar.of_connectToServer() = -1 then
     messagebox("ERROR", "Could not connect")
     halt close
   end if
```

To use a proxy, you define it as a variable (just like any other NVO) and create it, but since it’s CORBA you can’t just use the CREATE. Here’s what you do:

```
   n_jaguarObject = local Object
   go_jaguar.createInstance(local Object)
```

Now you can use your proxy as if it was a PowerBuilder NVO.

It’s sort of like the paging we used to do years ago when writing word processors in C. Remember that? You’d read a part of the file that was edited and try to keep your current index somewhere in the middle of that to minimize the “hicups” the user would see when doing page down and up. It’s pretty much the same concept.

The shared objects don’t go out of scope, and there’s never more than one of these at a time in Jaguar memory, since that one is shared. It fires off a separate thread for every process that calls it.

The services objects allow you to create something akin to Windows services or UNIX CRONs, except that it runs entirely within the context of Jaguar. It fires off and just keeps running forever, sleeping for a programmer-defined period after each run. There’s a function call to make it sleep (JagSleep).

So these are the types of components. The story doesn’t end here though. Inheritance is fully supported so you have all the power of inheritance in your components. Mine, for example, I created a base class and then added code that would write to the server log file. Of course, that wasn’t quite enough (when is anything enough?). I read an article in PowerBuilder Developer’s Journal in which Mike Barlotta described how to use Jaguar component properties. In this way, whoever is charged with maintaining Jaguar can fire up an application called the Jaguar Manager, find a package and a component within that, then right-click on it and add a new property. The property, like any other, has a name and a value except that it’s a string. From within my component I can get that property with a function called getContextKeywords. It returns me to an array of strings that are the values for all the properties the Administrator set up by a particular property name in the Jaguar Manager. I try to get the Administrator never to set up more than one, but he or she may set up a property named “foo” on more than one component, in which case I get one array element for each of them.

I have a ServicesManager object that’s really just a shared object, not a service at all. Basically, the idea is that the programmer can register (taking a page from PFC) objects that he or she inherits from a component I created with this manager. I’ll store the objects in an array and check them periodically. If, from the internal properties of the component, it finds one that’s ready to fire, it fires the component run method and resets the next time it’s due to run.

I’ve got this set up so that each runnable component can be set up to run on specific days of the week, days of the month, or every day. The programmer can also specify a start and end time along with an interval at which to run the component run method.

Author’s comment: This is a message that I sent to Cliff High, an old friend and fellow programmer. We’ve worked together from time to time for the last two decades – mostly with C++ contracts – but recently with a PowerBuilder contract. I helped him learn PowerBuilder through e-mail and he was strongly impressed with the language, especially the ability to create truly object-oriented applications. This contract wasn’t EAServer so he wasn’t exposed to it. The message above is in response to his asking me how I like my current EAServer contract.
In other words, a programmer can inherit from my component, write some code in the run method, then specify that this code will run every hour on the third of each month; every Monday, Wednesday, and Friday; or anything he or she chooses. I even have it set up so the programmer can tell the component to run only on the third day of specific months.

I’ve also set up this manager so that it either perpetually loops (with a yield in there, of course, so it doesn’t monopolize the server) or it will, at the end of each run, find the next time that any of its components need to run and will “sleep” until that time. The problem with this is that you can’t register objects with it while it’s sleeping. So during development I just loop. Once it goes into production we can set another component property to make the thing sleep and be better behaved.

Pretty cool, huh? Now the programmer doesn’t have to call the JagSleep, and I get to handle the sleeptime entirely outside the component. I can optimize the sleeptime at runtime by just going into the Jaguar Manager and changing the value of the property. Of course, it’s not perfect. Let’s say that your sleeptime is 10 minutes and your system is experiencing heavy traffic, so you want to make it 20 minutes. Let’s also say that your component just finished its run. If you change the 600 to 1,200, then your component is still going to wake up after 10 minutes, not 20. But after that run it will go to 20-minute intervals. I guess that’s not quite as bad as the other way around. If it’s running every 10 minutes and you want to change that to every minute, then you have to wait the 10 minutes after the change before it goes to every minute. I guess you just can’t have everything.

I mentioned server logs before. When I read Mike Barlotta’s article I got curious and started exploring all the other component properties that were already there. I found one that told me if I was running Jaguar in debug mode (oh yes, there’s a remote debugger for Jaguar components, but that’s worth a whole message in itself). So I went to my logging function in my ancestor component and put in the code to find out if Jaguar was in debug (again using the getContextKeywords method). I set it up so that the component would log only if we were in debug.

One more thing before I sign off and write some more Jaguar components. Mike Barlotta wrote two books on Jaguar (one, Taming Jaguar, which he coauthored, came out in August) in which he describes something called a State Manager. This is a component that lets browser-based applications store state information on Jaguar. I expanded this a little by adding a timeout on each variable and then writing the state information in a table. I then created a Garbage Collector (a fancy name for a typical services component) that wakes up every 600 seconds (or whatever the Administrator defined for StateGarbageCollectorInterval). It runs through the table removing expired data.

I’ve skipped the whole topic of the HTML DataWindow. Basically there’s new functionality in the DataWindow that allows you to return HTML to a browser that will paint the DataWindow in the browser – complete with presentation styles and JavaScript code to handle events and validation, and basically run the DataWindow.

Sorry I’ve given such a simplistic explanation of the new Jaguar, but hey, Mike Barlotta wrote a whole book on the subject and still found that he had to write another.
**PowerBuilder News**

All things of interest to the PB community

**PBL Peeper**

**8/05**

Techno-Kitten.com announced the release of the lastest version of PBL Peeper. Version 1.3.1 includes a number of enhancements and bug fixes and can be downloaded from their Web site.

[www.techno-kitten.com](http://www.techno-kitten.com)

**PBBrowser Beta**

**8/18**

OOWidgets announced the availability of a beta version of PBBrowser. The beta includes a number of new features and bug fixes and can be downloaded from their Web site.

[www.oowidgets.com](http://www.oowidgets.com)

**Watcom C++ and Fortran Open Sourced**

**8/22**

Sybase announced that it plans to release the source code for the Watcom C/C++ and Watcom Fortran compilers under an Open Source license agreement. This move will allow existing customers to continue to evolve the Watcom compiler products for their own use.

[www.openwatcom.org](http://www.openwatcom.org)

**OEM Agreement with Teknowledge**

**8/07**

Financial Fusion, Inc., a subsidiary of Sybase, and Teknowledge Corp., announced the signing of a worldwide original equipment manufacturers (OEM) agreement to provide account aggregation to leading global financial institutions that demand advanced functionality. As part of this partnership, Financial Fusion will use Teknowledge’s TekPortal software as the financial information aggregation component of its Web & Wireless System.

[www.teknowledge.com](http://www.teknowledge.com)

**Gauss Enterprise Joins e-Portal Alliance**

**8/07**

Gauss Enterprise of Germany became a member of the Sybase e-Portal Alliance. Their VIP product provides for content management that allows users without specialized technical knowledge to easily embed all types of content.

[www.gauss-enterprise.com](http://www.gauss-enterprise.com)

**Sybase Joins Winter Database Scalability Program 2000**

**8/18**

Winter Corp. announced that Sybase became a member of the Sybase e-Portal Alliance. Their VIP product provides for content management that allows users without specialized technical knowledge to easily embed all types of content.

[www.wintercorp.com](http://www.wintercorp.com)

**SSPA Certification Program for Support Professionals**

**8/17**

The Software Support Professionals Association (SSPA) announced the development of the SSPA Certification Program for Support Professionals. The program, developed in conjunction with 40 companies representing the IT industry (including Sybase), will consist of support center industry-tailored training and certification testing.

[www.supportgate.com](http://www.supportgate.com)

**OEM Agreement with DATABASICS**

**8/22**

Sybase announced that it reached an OEM agreement with DATABASICS, Inc., to embed EAServer software in DATABASICS’ TimeSite and ExpenSite products. TimeSite and ExpenSite help companies track employee and project profitability via the Web. TimeSite PSA is an electronic time sheet software product, whereas ExpenSite is an expense tracking program.

[www.data-basics.com](http://www.data-basics.com)

**Enterworks Joins e-Portal Alliance**

**8/30**

Enterworks announced that it has joined the e-Portal Alliance. Enterworks Content Integrator powers the full range of e-business interactions by managing content in real time from disparate sources and consolidating it into a single Web-enabled view.

[www.enterworks.com](http://www.enterworks.com)

**Magnet Communications**

**8/30**

Financial Fusion, Inc., a subsidiary of Sybase, and Magnet Communications announced that UniversalOFX, a key component of the Financial Fusion Server, will be integrated with Magnet’s Internet business banking products. This integration allows Magnet end users to exchange data with financial management software.

[www.magnetbanking.com](http://www.magnetbanking.com)

**Lifescape Receives Innovative Solutions Award**

**8/30**

Sybase presented Lifescape with an Innovative Solutions Award for its impressive use of EAServer to improve business-to-business connectivity via the Internet. The award, presented to Lifescape during the opening ceremonies of TechWave 2000, recognizes the degree of technological sophistication of the LifeNet health care messaging platform.

[www.lifescape.com](http://www.lifescape.com)

**PowerDesigner 7.5**

**8/30**

Sybase announced the general availability of PowerDesigner 7.5. The new version provides a single tool that allows both object modeling and data modeling techniques with full repository capabilities. PowerDesigner 7.5 features improved Unified Modeling Language (UML) class diagram modeling and an enhanced repository for refined enterprise modeling.

**Sybase Begins Move to Dublin, California**

**8/31**

Sybase broke ground on the company’s new corporate headquarters and campus in Dublin, California. The groundbreaking is the first step toward the construction of the new corporate facilities, scheduled for completion by June 2001.

bruce.armstrong@teamsybase.com
SYS-CON Media’s Java Developer’s Journal Becomes Fastest Growing Technology Title on Newsstands

Newsstand Distribution of JDJ, the World’s Leading Java Resource, Expected to Triple in 2001

(Montvale, NJ) – SYS-CON Media, Inc. (www.sys-con.com), headquartered in Montvale, New Jersey, has announced that the world’s leading Java resource, Java Developer’s Journal, is set to become the fastest-growing technology magazine on U.S. newsstands and throughout the English-speaking world.

“We have developed a progressive growth program for selected magazine titles that show future strong single-copy sales potential and Java Developer’s Journal is one of several titles we have chosen for that program,” said Robert Castardi, president of Curtis Circulation Company. “Single-copy sales of Java Developer’s Journal in the United States are up an impressive 47% for the first six months of 2000.”

“We’re committed to bringing the high-quality editorial content of JDJ to Java professionals around the globe,” said Fuat Kircaali, president and CEO of SYS-CON Media and publisher of Java Developer’s Journal. “We’re very pleased to see an impressive sales growth for JDJ on the newsstands as well as our subscriber base since JDJ’s first issue five years ago.”

During the first six months of 2000, Curtis distributed 96,393 copies of Java Developer’s Journal to specialty sales outlets to sell 68,291 copies – an efficiency of 70.8% – a success ratio not seen in any other technology title for the same period. Specialty store sales such as Barnes & Noble and Borders Bookstores showed an increase of 152.6% as compared to the same period of 1999. Curtis also distributes JDJ via its wholesalers and to independent bookstores.

Java Developer’s Journal’s BPA audited circulation has consistently exceeded the combined circulation of all other Java-related magazines for the past five years.

JDJ’s worldwide newsstand distribution is managed by Curtis Circulation Company, the largest magazine distributor in the world. The New Milford, New Jersey-based firm is a division of Hachette Distribution Services, headquartered in Paris.

About SYS-CON Media

What Cost Optimization?

To speed or not to speed

Written by Michael Deasy

It may not be economically feasible (or necessary) for you to throw money at speeding up applications. Spending a certain amount of money on speed may be expedient and eventually even necessary, but it shouldn’t be our first choice as analysts or managers. Too many hours are spent working up cost justification analyses and AFEs (Authority for Expenditures) that could have been spent on actual repairs.

You could finance the Sys Admin people upgrading your UNIX server at an often-huge cost or get speedier desktop machines for your users (implying faster solitaire). You could even get a contractor to come in. While all these will probably speed things up, you could roll up your sleeves, put on your work boots, save the company some serious bucks, and win the adoration of your users as you turbocharge their apps.

The following three ideas cost only your time and intellect. The combined effect of these alone can result in a significant performance increase.

Your Mamas and Your Sisters Don’t Work Here

This sage advice was given to me while contracting with a research firm in Huntsville, Alabama. It was in reference to empty pop cans, but that’s beside the point – which was to clean up after yourself. It’s vital that we do our house cleaning and examine the timing of our user complaints.

Does the application seem to slow down more as the day wears on? This typically means that you haven’t done the necessary housekeeping. There is a memory leak in your code. You can fix this by looking for easily correctable mistakes.

Those of you using a version of PB previous to 6.0 may have trouble if you aren’t doing the proper housekeeping. There are some simple maxims you should always remember: if you created it, destroy it; if you opened it, close it; if you connected, disconnect. This may seem an oversimplification, but it will serve you well. You also need to go ahead and upgrade to a supported version of PB.

Post-6.0 PowerBuilder will employ Garbage Collection and clean up most of the orphaned or circular references in your code. There are areas that this doesn’t affect, however. Sybase has well documented that they don’t Garbage Collect on Visible, Timing, or Shared Objects. Also, if you have posted events or functions, they won’t be GC-Collection until after the event or function has run.

The Realized Pain of the Few vs the Wails and Moans of the Many...

As the number of users and data grows, practical changes must be made to accommodate scale. It’s vital to examine where in the course of data entry or the job stream you’re doing the heaviest processing. Is every user sitting through error checking?

I support an application that had a lengthy process doing extensive error checking. The process was positioned such that a one hundred-plus people had to watch a spinning hourglass or (my favorite cursor) the burning dollar bill while it ran numerous times daily. After examining the functionality involved, it became clear that I could move the error checking into a different area and still effectively suppress bad data entry. I moved that checkpoint to a place where a small group could confirm the validity of the information so the large group never had to see it.

This change gained hours a day in user time, and we didn’t lose any quality assurance.

Too Many Stops Along the Way

How many server hops is the application making in the course of touching the network or the database? In the course of testing applications, I once found some users on a given floor were making multiple hops to get from their localized network to the servers where the database lived. This was awkward, but not a difficult task for the network people to correct.

They did some simple testing to check the jumps from my machine to the server using TraceRT, then went to the users and confirmed that they had the same problem. Once it was established that there were n-jumps to get to the servers, they began adjusting the connectivity to give us a direct line to the database server in question. This simple and inexpensive change shaved minutes off the Save process for deal entry.

And the Final Word

There are many ways to increase system speed and optimize your application. First, as a developer make every effort you can to ensure that your code is the best it can be, that your connectivity is not in question, that the DBAs have the database well tuned, and that you don’t have any sloppy SQL. Then you can begin to look outward. You’ll find that your users will treasure your efforts as they run their jobs in a shorter and shorter time frame.

Author Bio

Michael Deasy is a staff systems analyst for the Williams Companies in Tulsa, Oklahoma. He holds an MBA from Southern Nazarene University. He has been working with PowerBuilder since v3 and is president of the Tulsa PowerBuilder Users Group.

mike.deasy@williams.com

www.PowerBuilderJournal.com