Issue three | November 2003

Generation PB

PowerBuilder Newsletter

Welcome to the third issue of Generation PB

This issue is packed with news and events from the Sybase PowerBuilder team.

What has happened since the last one? A long hot summer for most of us, during which the annual Sybase User Conference – Techwave was held in Orlando, August. At Techwave the long-awaited Pocket PowerBuilder 1.0 was launched and the latest PB roadmap was unveiled covering the forthcoming features and functionality of PowerBuilder.

As PB 9.0 marked the foundation of UDE (Unified Development Environment), future releases will reinforce Sybase's commitment to deliver a tool that keeps up with emerging technologies such as Web and component based development, modelling and life-cycle management (with PowerDesigner), Web Services, Wi-Fi and application integration.

You will read how you can access diverse components from your Pocket PowerBuilder (PPB) application and discover another new PowerBuilder product, DataWindow.NET which will start its beta cycle shortly, as well as the latest product news and events. See PowerBuilder in the Press - I will be having interviews about PowerBuilder in the IT Press throughout Europe.

The Future is Brighter for PowerBuilder !

Steven Dunn - *EMEA PowerBuilder Sales Manager*

Sold out: PB 9 Launch European tour

Cordiali saluti da Milano - Sybase Italy PB9 launch seminar

Saludos de Barcelona - AST's PB9 launch seminar

Grüße von Hamburg- PowerPeople's PBUG seminar

What an amazing journey it has been: 12 events covered in 2 months with 900+ attendees in total.

We want to take the opportunity to thank the organisers, the presenters and of course you, the PowerBuilder users, for making it such a successful event.

So now the question is: have you practised what our technical evangelists preached?

We hope that the product demonstrations (Web services, XML DataWindow, PBNI, Pocket PowerBuilder) benefited you and your projects.

We look forward to seeing you again soon. Check out page 6 for our forthcoming events.

In the next issue:

More on PowerBuilder, Pocket PowerBuilder, DataWindow.Net features.
To subscribe, email us at **softwarecentre@sybase.com**

Who's who? Cao Lin

Let us introduce Cao Lin, Senior Engineering Manager.

Cao Lin
Senior Engineering Manager

Dr. Cao Lin holds a MEng and a Ph. D. His software development experience includes banking database system, decision support system, C++ code generator and information retrieval system etc. Based in Singapore he has been working on PowerBuilder development and management over 5 years. He started from PowerBuilder 6.5 development for Chinese and Korean versions at Sybase. He built up the excellent PowerBuilder Asia development and QA teams in 2000. Together with the North America PowerBuilder team, he led the successful release of PowerBuilder 8.0. As the senior engineering manager, he is leading the worldwide development for PowerBuilder. Dr. Cao Lin and his team had made PowerBuilder 9.0 the most exciting release after PB 5.0 release. He is encouraged with seeing PowerBuilder 9.0 is restoring customer confidence.

He envisions PowerBuilder as next-generation application development tool that abstracts and simplifies the complex technologies for existing and new developers to be more productive and competitive for application development on open platforms. In particular, the following key features are planned to be in PowerBuilder 10 or subsequent releases. The iterative design-development approach will allow developers to model and reverse-engineer to see a complete picture of the class diagram. The DataWindow.NET will allow developers to use DataWindow with other .NET languages. The .NET compiler will allow PB developers to take advantage of other .NET languages in PowerBuilder IDE. The DataWindow for Java will allow Java developers to use DataWindow in Java platform. In the future, PB developers would have options to generate PB, .NET or Java applications in PowerBuilder IDE for deployment. Developers may also choose to deploy their applications to desktop or wireless devices.

Tips and Tricks

10 Tips For Migrating Applications to PowerBuilder 9.0

by Lee Cheetham, Sybase Professional Services Consultant

1) Use the Migration Assistant to identify obsolete functions and events. The Migration Assistant scans PowerBuilder libraries to detect potential problems with syntax prior to migrating the application. Obsolete Syntax is available from version 6 onwards. The Migration Assistant can be found on the "Tool" tab of the "New" object dialogue box.

2) PSR files, if an application attempts to open a PSR file created in an earlier version of PowerBuilder (pre 8.0 build 7063 or pre 7.0 build 10102) it will fail. This is due to a change in the SaveAsASCII function. To fix this problem, recreate the PSR in a later build of PowerBuilder.

3) New reserved words (Try, Catch, Finally, Throw, Throws) were introduced in PowerBuilder 8.0 for exception handling. The Migration Assistant will identify any occurrences of these reserved words. To fix this problem, change all the instances of "Try, Catch, Finally, Throw, Throws" in your application to alternative names using a previous version of PowerBuilder.

4) Distributed programming support has been discontinued since PowerBuilder 8.0. The "Transport" object has become obsolete. The application events "connectionbegin" and "connectionend" no longer get fired. The solution is to deploy your server side NVOs to EAServer.

5) SystemError Event processing has changed between PowerBuilder 7 and the later versions of PowerBuilder. In PowerBuilder 7, when an error occurs the "SystemError" event is triggered, after the "SystemError" event processing has completed, control is returned to the script where the error occurred. In PowerBuilder 8 and 9, when an error occurs the call stack is unwound, then the "SystemError" event is triggered, after the "SystemError" event processing has completed, processing does **NOT** continue at the point where the error occurred. Catch and handle errors locally using the new exception handling functionality available in PowerBuilder 8 and 9.

6) IsValid function was changed in PowerBuilder 8 to return "FALSE" if the object is not instantiated or an invalid object has been passed, prior to version 8, if an invalid object was passed the "SystemError" event would be triggered.

7) Windows no longer inherit the Application Icon, you must now code or set the property to **"AppIcon!"**

8) TreeView Event, the **"rbuttonup!"** event no longer fires, this was introduced in PowerBuilder 7.

9) ListView, the "**pbm_rbuttonup!"** event no longer fires when rightclicking on a ListView item, but it does fire if you click on the white area in the ListView. The solution to this is to use the new **"pbm_contextmenu!"** event, which always fires when the right mouse button is released.

10) Web Targets, that use the WebDataWindow control must be modified to use the new HTMLGenerator90 component. Right click on the Web DataWindow Design Time Control on your web page and select properties. On the "HTML Generator" tab change the reference from HTMLGenerator80 to HTMLGenerator90. After changing the HTMLGenerator, the database connection properties are sometimes lost, so it is important to check the "Connection" tab and ensure that the database connection setting is still correct after the change.

DataWindow .NET – Bringing the Power of the DataWindow to Visual Studio .NET

By David Avera, Staff Engineer, Sybase, Inc

Experienced PowerBuilder programmers know well the power and capability of the DataWindow control and the DataStore. This patented (US patents 5566330, 5752018, 5832481, 5937415) software is described as an "Invention that features a system for defining a database interface that allows an applications programmer to graphically define, display, and use [a visible object] the data window to indirectly manipulate data in an application database". Secondly, it says "the invention features a program object [a DataWindow object] providing an interface between a computer database manager for managing a database table and a client application program external to the interface object."

One can read the patent applications online as a cure for insomnia. Suffice to say, the DataWindow is the crown jewel in PowerBuilder, one of the major elements that has set PowerBuilder apart from all of its competitors. To date, no one else has created a product that rivals the DataWindow in capabilities and ease of use.

Why DataWindow .NET?

The DataWindow has been exclusive to PowerBuilder development since the early 1990's, with the exception of the DataWindow ActiveX and the DataWindow for Java ™ controls. Microsoft's .NET technology has offered Sybase an opportunity to extend the DataWindow franchise to developers using Microsoft languages and tools. This is a very large marketplace, including Visual Basic developers (primarily), but with an increasing number of C# developers.

In addition, an increasing number of PowerBuilder shops have mandates to create some of their applications with .NET tools. DataWindow .NET will allow these shops to maintain their investment in DataWindow technology – all without having to learn and perfect the use of Microsoft's DataGrid, DataView, DataSet and the various types of adapters and connectors.

Up until now, users of Microsoft development tools have had a hodgepodge of components such as the DataGrid, DataView, DataAdapters for each supported database, DataSets, Sql Connection and Sql Command objects. A vigorous third party market exists that extends these components, making them easier to use (and more like the DataWindow).

Sybase will offer a better solution for data access, reporting and manipulation-DataWindow .NET.

The DataWindow .NET product will provide the full services of the familiar PowerBuilder DataWindow and DataStore components in the Visual Studio IDE. Users will be able to program DataWindow .NET with any .NET language.

What makes up DataWindow .NET?

DataWindow .NET will be delivered as a set of DLLs that constitute a .NET front end and a back end DataWindow server. To create and maintain DataWindow design time definitions, the DataWindow Builder will be part of the package.

The DataWindow .NET front end provides the interface between the Visual Basic or C# application program and the DataWindow server. The front end maps the familiar methods, events, and properties of the DataWindow to the application. The front end also ensures that .NET applications will only directly interact with pure .NET code. The back end DataWindow server is essentially invisible to the application; its only requirement is that the DLLs be located on the load path.

The DataWindow server manages the loading and presentation of DataWindow objects and manages communication with databases. The traditional functionalities of the DataWindow, such as maintaining the data buffers, sorting, filtering, exporting data to a variety of file formats and DataWindow expressions, are all processed here.

How do you use DataWindow .NET?

Let's get to the good stuff!

Assuming that you have certain DataWindow definitions stored in a PBL or PBD file, let's create a simple C# application using the Visual Studio New Project wizard. Select "Visual C# Projects" and "Windows Application". We will use a DataWindow in the application, connect to the database and retrieve and display the data. We will also add an event handler for the DataWindow RetrieveEnd event.

Author's note: The following images and code are from the pre-alpha version and are not guaranteed to be exactly the same in the finished product.

Figure 1. The initial form display with the Toolbox

As you can see in the Toolbox, there is a DataWindowControl component. Click on this item and drop it on the form. The result is a blank rectangle identifying the location of the DataWindowControl.

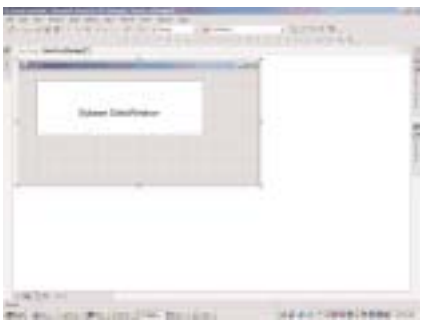


Figure 2. The form displayed with an empty DataWindow Control

Fill in some DataWindowControl properties

Figure 3. *The DataWindowControl Properties tab sheet.*

Click on the DataWindow to make it the current control and bring out the Properties tab.

Here we can see some familiar DataWindow object properties. Select "LibraryList" to define where the DataWindow definition is to be loaded from. Then select the DataObject property to select a specific DataWindow object.

Select a PBL (or a PBD) containing DataWindow definitions

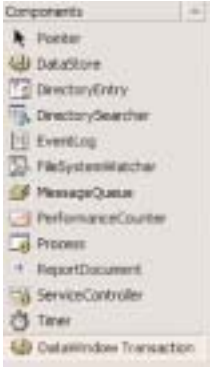
Figure 4. *The LibraryList dialog box.*

Select a DataWindow from the PBL or PBD

Figure 5. *The DataWindowObject selection dialog box.*

The DataWindowControl on the form now looks like this:

Figure 6. *The DataWindowControl with headers*



Add a DataWindow Transaction to the application.

Figure 7. *Visual Studio Toolbox*

From the Components tab in the Toolbox select the DataWindow Transaction component and drop it on the form and fill in its properties

Figure 8. *Transaction properties.*

(Cont. Fig.7 & Fig.8) Dragging and dropping the transaction, setting its properties will instantiate a transaction object (we will call it SQLCA) automatically in the application. This of course may also be done in the C# code. If you examine the code generated by the IDE you will see:

```
this.DWC1 = new
Sybase.DataWindow.DataWindowControl();
this.SQLCA = new
Sybase.DataWindow.Transaction(this.components);
```

and

```
this.DWC1.DataWindowObject = "d_dept";
this.DWC1.LibraryList = "E:\\PB Diamond
Workspaces\\externa.pbl";
```

Now for some actual coding

In the form constructor, you can issue the connect to the database:

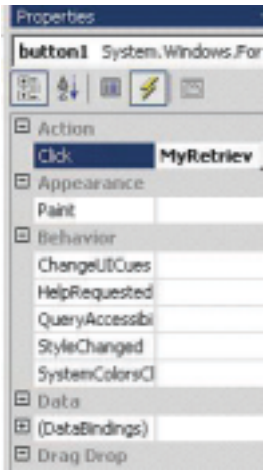
```
public Form1()
{
      //
      // Required for Windows Form Designer
support
      //
      InitializeComponent();
//
// TODO: Add any constructor code after
InitializeComponent call
      //
      SQLCA.Connect( );
}
```

Now let's code for the DataWindow retrieval. Select a Button control from the Windows Form tab on the Toolbox and drop it on the form. Add some properties, like "Retrieve DataWindow" for the button text, select events in the properties tab and enter the name of a method you wish to use.

Figure 9. *The form with a button.*

Figure 10. *The button properties tab sheet showing the clicked event*

The IDE will take you to the form's code view with the following boilerplate code inserted:



Cont. Fig.10

```
private void
MyRetrieve(object sender,
System.EventArgs e)
{
}
```

Insert the following code into the MyRetrieve method:

```
DWC1. SetTransObject
(SQLCA):
DWC1. Retrieve ( );
```

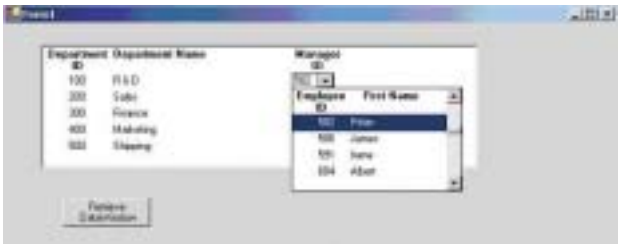
```
intDataWindowControl.Retrieve (params
object[] Argslist)
Retrieves rows from the database for the
DataWindow control. If arguments are included,
the argument values are used for the retrieval
arguments in the SQL SELECT statement for the
DataWindow object.
```

This demonstrates the Visual Studio intellisense ability to put up documentation for a method by hovering the mouse over the verb.

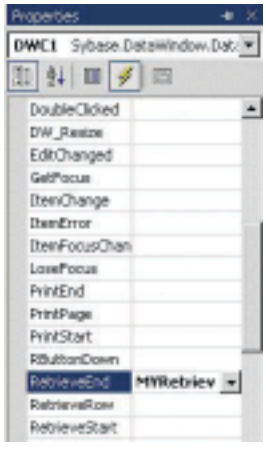
Finally, run the application and press the "Retrieve DataWindow" button:

Figure 11. *The form displayed at runtime*

The familiar Department table has been retrieved and displayed. Note that Drop down DataWindows work.



Suppose we want to know when the retrieve ends and how many rows were retrieved



Return to the IDE, select the DataWindowControl to make it the current control, access the properties tab.

Figure 12. *The DataWindowControl property tab sheet showing events.*

You will be taken to some boilerplate code in the code view:

```
private int MYRetrieveEnd(object sender,
Sybase.DataWindow.DataWindowEvents.RetrieveEndArg
s args)
{
      return 0;
}
```

Add the following code to the method:

```
String Rows = args.RowsRetrieved.ToString( );
MessageBox.Show ("Rows Retrieved = "+ Rows,
"Retrieve End");
```

Retrieve with a RetrieveEnd event handler method

Figure 13. *The form executing with a Retrieve End messagebox*

This example demonstrates how easy it is to utilize an existing DataWindow in a .NET C# application. It is just as easy to do this in Visual Basic .NET.

Beyond the simple example above, we will provide the full power of the DataWindow for .NET programmers. One of our internal PowerBuilder experts is porting the PowerBuilder DataWindow examples to Visual Basic with good success for a product in the pre-alpha state. These examples will demonstrate that more complex usage of the DataWindow will work as well as the simple example.

What will be supported in the final product?

Almost all the familiar DataWindow methods, properties and events will look just as they do in PowerBuilder with few exceptions:

1. DataWindow events will be coded using the .NET event model, the same event arguments will be available but they will be accessible in specific to the event argument objects.
2. Dot notation is not supported in DataWindow .NET. SetProperty and GetProperty methods will be provided.
3. The Richtext DataWindow style is not supported.

Summary

In this article I have tried to show you how easy it will be to utilize DataWindow .NET in a Visual Studio application using C#. If you know Visual Basic, the process is just as easy. The example showed how to click and drop items from the Toolbox, specify some property values for these items, add a small amount of code and you have created an application that retrieves and displays database data. Behind this simple example is the full power of the DataWindow including:

● Data formatting ● Masked data entry ● SaveAs any supported file type including XML ● Powerful reporting options like groups and summary bands – A variety of reporting formats, grids, free form, cross tabulations, graphs and more. ● Updating the database, including stored procedures ● DataWindow expressions ● Sorting ● Filtering ● and more, a lot more.

With DataWindow .NET, Sybase offers Visual Studio developers an alternative from the DataGrid, the DataView and other Microsoft objects. For the PowerBuilder programmer who must use .NET technology in an application, Sybase offers a means to leverage their DataWindow experience in the new environment.

New hot off the press!

Forthcoming events: mark your calendar!

November 18th 2003 – **Norway: Sybase Seminar.** For more info & to register, visit **www.sybase.no**
December 3rd 2003 –**UK: STUN conference.** For more info & to register, visit **www.stun.org.uk**
December 4th 2003 – **Denmark: PBUG meeting.** For more info & to register, visit **www.ravenholm.dk**
December 8th 2003 – **The Netherlands: Sybase User Group.** For more info & to register, visit **www.dsug.nl**
December 9th 2003 - **Belgium: Sybase Techserie, Kinepolis Brussels.** For more info & to register, visit **www.sybase.be**
December 10th 2003 - **Luxembourg: PB 9 seminar, Sofitel Luxembourg.** For more info & to register, visit **www.sybase.be**

New PB 9 books:

Two new books on PowerBuilder 9.0 covering **Advanced Client/Server Development** and **Distributed Development** are available, written and edited by experts in the field.
PowerBuilder 9 Advanced Client/Server Development by Bruce Armstrong, Millard F. Brown III
PowerBuilder 9 Internet and Distributed Application Development by William Green, John D. Olson

Also available **Mobile and Wireless Design Essentials** by Martyn Mallick, iAnywhere Solutions Product Manager
Mobile and Wireless Design Essentials is the first and only book that provides a detailed analysis of the complete range of current mobile and wireless technologies from a software developer's perspective.

End Of Life Notification for PowerBuilder 7.0.x.

We encourage you to upgrade your software to PowerBuilder v8.0 or v9.0 (Enterprise, Professional and Desktop) since PowerBuilder v7.0.x (Enterprise, Professional and Desktop) is now end of engineering support.

Free evaluation CD

Call us TODAY to purchase your PB & PPB licences or to get your FREE evaluation CD or alternatively visit us on www.softwarecentre.sybase.com

France - 01 41 91 96 80
Germany - 069 9508 6182
UK - 0870 240 2255
Italy - 02 696 820 64
Switzerland - 01 800 9220

Spain - 091 766 46 00
Advanced Software Technology
Belgium - 03 825 07 95
Formula Opensoft
Netherlands - 078 673 6341
Formula Opensoft

Norway - 0230 555 00
Ravenholm Group
Sweden - 013 31 15 88
Linsoft
Denmark - 044 88 99 00
Ravenholm Group

Current release/EBF info

DESCRIPTION	VERSION	DATE	TYPE
PB Enterprise - 9.0.1 (6533) Maintenance Release	9.0.1	05 Sep 2003	Maint/Update
PB Professional - 9.0.1 (6533) Maintenance Release	9.0.1	05 Sep 2003	Maint/Update
PB Desktop - 9.0.1 (6533) Maintenance Release	9.0.1	05 Sep 2003	Maint/Update
PB Enterprise - 9.0.1 (6533) Localized Runtime Files	9.0.1	08 Oct 2003	Maint/Update
PB Enterprise - 9.0.1 (6533) Localized PFC Files	9.0.1	08 Oct 2003	Maint/Update
PB Enterprise - 8.0.4 (10501) Maintenance Release	8.0.4	06 Oct 2003	Maint/Update
PB Professional - 8.0.4 (10501) Maintenance Release	8.0.4	08 Oct 2003	Maint/Update
PB Desktop - 8.0.4 (10501) Maintenance Release	8.0.4	08 Oct 2003	Maint/Update
PB Enterprise - 7.0.3 (10312) EBF Release	7.0.3	20 Aug 2003	EBF/Patch

Note: PB v7.x EBF build is the final deliverable for this codeline which was EOLd on 7/15/2003:

Note: You can find more information on www.downloads.sybase.com

Pocket PowerBuilder, SOAP & PocketSOAP

By Ian Thain, Technical Evangelist, IPG, Sybase Inc.

So you may have diverse components in your enterprise and need to access them from your Pocket PowerBuilder application? I will not waste too much time on SOAP & WebServices, as there are lots of reading that you can do on those subjects, especially if you can't sleep one night. All I will do now is to set the scene around the technologies of SOAP & WebServices and then let's see what we can do.

SOAP

Simple Object Access Protocol (SOAP) is widely viewed as the backbone of a new generation of cross platform, cross language, distributed applications – WebServices. As its name suggests, it is a lightweight, XML based protocol for the exchange of information in a distributed environment. The SOAP protocol consists of three parts. An **envelope** that defines a framework for describing what is in the message and how to process it. Next a set of **encoding rules** for datatypes, and finally a **convention** for remote procedure calls & responses.

WebServices

A WebServices public interfaces and bindings are defined and described using XML, and are identified by a URL. The definition and description are defined using WSDL – Web Services Definition Language and can be discovered by other software systems via UDDI. Once discovered interaction takes place using XML based messages conveyed by Internet protocols (SOAP), in a manner defined by its definition.

PocketSOAP

PocketSOAP is an Open Source SOAP client Component Object Model (COM) component, originally targeted for the PocketPC, that is now developed for Pocket PC and Windows 95, 98, Me, NT4.0 and 2000. PocketSOAP can make remote procedure calls to SOAP servers implemented with 4s4c, ROPE, Apache SOAP, SOAP::Lite, DM's SOAP/Perl and the XMethods SOAP Server. The package includes a Hypertext Transfer Protocol (HTTP) client for making HTTP based SOAP requests, but other transports can be added. PocketSOAP is distributed under the Mozila Public License (MPL). PocketSOAP includes the following features:
SOAP Section 5 encoding support, support for document/literal style SOAP services (such as ASP.NET), attachments support via both DIME and SOAP with Attachments, Support for SOAP, HTTP 1.1 support including persistent connections, SSL, proxies, authentication, proxy authentication, redirects, cookies and compression. There are many more that can be viewed at **http://www.pocketsoap.com**

Pocket PowerBuilder & PocketSOAP

Pocket PowerBuilder & PocketSOAP interact through PPB's Interface for PocketSOAP. This is an external DLL wrapper (since PPB does not have a COM object aka OleObject), which provides a simple access to the PocketSOAP package, via its COM Interface. The model for the API is that first the service object is created and that returns the access handle. Then you can set the various attributes in whatever order you desire. Internally, the attributes are simply stored for later use. Then you make the SOAP call. This uses the previously set attributes. Finally, you destroy the service object, which cuts away all the previously created COM objects. All this is built on top of the PocketSOAP system. So to run with PocketSOAP, retrieve the installs from **http://www.pocketsoap.com** and install them on both the desktop and Pocket PC device.

EAServer

Let us look at an example of using PocketSOAP within Pocket PowerBuilder to call a WebService on EAServer 4.2. This WebService is in fact an EJB that retrieves old stock prices from an ASA Database via a connection cache. This EJB is exposed as a WebService by defining it through the Web Services Toolkit within EAServer. This allows us to expose any component in EAServer as a WebService. The WST will generate all the WSDL needed.

This is the code that is needed to access the WebService through PocketSOAP.

```
string sMethod = "getPFShareList"
string sEndPoint  =
"http://localhost:10000/WEBSERVICES/SOAP"
string sNameSpace = "PFShareListAll/PFShareList"
int    cRetBufLen = 128
string sArgs
string sResult
long   lHandle
int    iRet

// create & set attributes on for the soap call
iRet = PocketSoap_Create( true, REF lHandle )
wf_message( iRet, "PocketSoap_Create")

iRet = PocketSoap_SetEndPoint( lHandle, sEndPoint )
wf_message( iRet, "PocketSoap_SetEndPoint")

// preallocate the result string
sResult = Space( cRetBufLen )

// make a "simple call"
iRet =  PocketSoap_SimpleCall( lHandle, sNameSpace,
sMethod, "", "" )
wf_message( iRet, "PocketSoap_SimpleCall")

// get the Result
iRet =  PocketSoap_GetResult( lHandle, cRetBufLen, REF
sResult )
wf_message( iRet, "PocketSoap_GetResult")

// clean up
iRet =  PocketSoap_Destroy( lHandle )
wf_message( iRet, "PocketSoap_Destroy")

wf_message (integer ai_ret, string as_method)
If ai_ret <> 0 Then
MessageBox("Return code", String(ai_ret) + " " +
as_method)
End if
```

The 3 important things we need to set are the **EndPoint**, **Namespace** & **Method**. The EndPoint indicates a specific location for accessing a service using a specific protocol and data format. The Namespace indicates the specific WebService and obviously the method is that method that we want to call within that WebService. In the example we use the method PocketSoap_SimpleCall which executes a synchronous call to the EndPoint URL with a single argument. This argument is a name/value pair and is optional. For a more complex call there is another method called PocketSoap_Call, which takes arguments as name/value pairs, but represented as a single string.

```
Name1 ~t value1 ~t data-type1 ~r~n
Name2 ~t value2 ~t data-type2 ~r~n
Name3 ~t value1 ~t data-type1 ~r~n
Etc
```

PocketSOAP will take care of generating the SOAP message & parsing the response. The SOAP call and the response from the server can be viewed by using the SOAP Util tool from Apache (http://www.apache.org) . This is started by using the command line

```
java org.apache.soap.util.net.TcpTunnelGui
10000 ithain-home 8080
```

This allows the tool to intercept all requests that are made to localhost:10000, display them then reroute them to ithain-home:8080. Like wise the tool intercepts the response from ithain-home:8080, display them and reroute them back to the caller.

For EAServer 4.x we need to set the **SoapAction** attribute. This is due to that fact that in EAServer 4.x we are using the SoapAction attribute, within our proprietary implementation of WebServices, though this is compatible with other implementation such as Apache. That being said other implementations also require SoapAction. No need to worry though as in EAServer 5.0 the implementation of WebServices will be **Axis** and then we need not set the SoapAction.

XMethods

Here is an example of calling a WebService on XMethods (http://www.xmethods.net). This uses a WebService that is a 20 minute delayed quote, implemented via GLUE. On XMethods there are many other WebServices implemented using MS .NET, Delphi, SOAPLite, WASP Server for Java, ColdFusion, BEA WebLogic, ApacheSOAP, AXIS & many more.

```
string sMethod = "getPFShareList"
string sEndPoint  =
"http://localhost:10000/WEBSERVICES/SOAP"
string sNameSpace = "PFShareListAll/PFShareList"
int    cRetBufLen = 128
string sArgs
string sResult
long   lHandle
int    iRet

// create & set attributes on for the soap call
iRet = PocketSoap_Create( true, REF lHandle )
wf_message( iRet, "PocketSoap_Create")

iRet = PocketSoap_SetEndPoint( lHandle,
sEndPoint )
wf_message( iRet, "PocketSoap_SetEndPoint")

// preallocate the result string
sResult = Space( cRetBufLen )

// make a "simple call"
iRet =  PocketSoap_SimpleCall( lHandle,
sNameSpace, sMethod, "", "" )
wf_message( iRet, "PocketSoap_SimpleCall")

// get the Result
iRet =  PocketSoap_GetResult( lHandle,
cRetBufLen, REF sResult )
wf_message( iRet, "PocketSoap_GetResult")

// clean up
iRet =  PocketSoap_Destroy( lHandle )
wf_message( iRet, "PocketSoap_Destroy")

wf_message (integer ai_ret, string as_method)
If ai_ret <> 0 Then
MessageBox("Return code", String(ai_ret) + " "
+ as_method)
End if
```

This time the SOAP Util tool was started using the command line

```
java org.apache.soap.util.net.TcpTunnelGui
10000 services.xmethods.net 9090
```

And the EndPoint was set to string sEndPoint = "http://localhost:10000/soap" rather than http://services.xmethods.net/soap in the code snippet above. You can also see from the request that this implementation does not require a SoapAction to be set.

Conclusion

Over the last few years we have been telling you to partition your applications, reuse code and take advantage of your favorite Application Server (EAServer). PocketSOAP can be used to access components exposed anywhere as WebServices and is accessible from Pocket PowerBuilder Applications via a PPB interface.

"This article originally appeared in PowerBuilder Developer's Journal (Vol. 10, issue 8)." To subscribe to PBDJ, register today at subscribe@sys-con.com, or for more information visit www.sys-con.com.