APPEON

Understanding the Architecture of Appeon 2.8

An Appeon Whitepaper

Appeon® for PowerBuilder®

July 2004

LAST REVISED: July 25, 2004

Table of Contents

1 Introduction

1.1 What the product can do

Appeon for PowerBuilder deploys entire PowerBuilder applications to the Web. PowerBuilder developers can build n-tier Web applications using only standard PowerBuilder client/server programming. N-tier Web applications can be generated from existing PowerBuilder application code, automatically. End-users are presented with the familiar rich user interface that keeps them highly productive.

From the native source code of PowerBuilder applications, Appeon for PowerBuilder can automatically generate bona fide n-tier Web applications. These Web applications precisely replicate the client/server user interface with HTML running in standard Microsoft Web browsers. The application business logic including DataWindows, NVOs, and Embedded SQL is deployed to a Java-based back-end. The back-end can be readily integrated with other J2EE, .NET, or Web Services-based Web applications. The standard n-tier Web architecture of Appeon for PowerBuilder offers the ultimate in scalability, availability, reliability, flexibility and security.

With Appeon for PowerBuilder, PowerBuilder enterprises can add a revolutionary new Web deployment option to PowerBuilder. Developers build Appeon for PowerBuilder projects, taking advantage of most commonly-used PowerBuilder features including the PFC framework. Then these projects can be deployed to the Web and client/server, at the click of a button. Whether building new Web applications or Webifying existing PowerBuilder applications, Appeon for PowerBuilder provides the fastest, most economical and lowest risk path to the Web.

1.2 Components of the product

Appeon for PowerBuilder consists of three major components or parts: Appeon Developer, Appeon Server and Appeon Server Web Component.

- **Appeon Developer** is a plug-in to the PowerBuilder IDE installed to the developer's PC. It provides a set of tools that enable the entire PB-to-Web conversion process to take place within the PowerBuilder IDE. These tools are accessed via a toolbar in the PowerBuilder IDE, which automatically loads each time PowerBuilder is started.

- **Appeon Server** is a set of J2EE CORBA and PowerBuilder CORBA server components that deploy to the application server. It provides the necessary run-time services for Appeon Web applications. These services include data connectivity, DataWindows support, n-tier NVO support, transaction management, PDF printing, and security.

- **Appeon Server Web Component** is a collection of JavaScript libraries that deploy to the Web server. It enables a PowerBuilder-style Web graphical user interface within standard Microsoft Web browsers. It is similar to the PowerBuilder Virtual Machine (PBVM) except that it is implemented in JavaScript so there is no client-side software to install and it weighs in at an ultra-light 1.2MB.

Table 1-1: PowerBuilder Virtual Machine Comparison

	PBVM	PKVM (PocketPB)	Appeon
Size	6-8MB[1]	3.5MB	1.2MB
Implementation	C DLLs	C DLLs	JavaScript
Client Installation Required	Yes	Yes	No

The popular MDI interface and windows, rich DataWindow functionality, menus/toolbars, and most controls are all available in the Web application.

1.3 Where the components reside

The Appeon Server and Server Web Component install to the n-tier Web architecture.

Table 1-2: Tiers in the Appeon Web architecture

Tier	Purpose	Appeon Component	Third-party Software
Client Web Browser	Runs Appeon Web applications.	None	Standard Microsoft Internet Explorer Web browser.
Web Server	Hosts the presentation layer of Appeon Web applications, responds incoming requests from client PC, and dispatches requests to the application server.	Appeon Server Web Component	Apache, IIS, Netscape, iPlanet, or EAServer.
Application Server	Hosts the DataWindows and n-tier NVOs of Appeon Web applications, and provides the necessary run-time services to the Web application, such as DataWindow retrieval and update, execution of business logic, security authentication,	Appeon Server	EAServer

[1] PBVM size depends on the number of database device drivers being used.

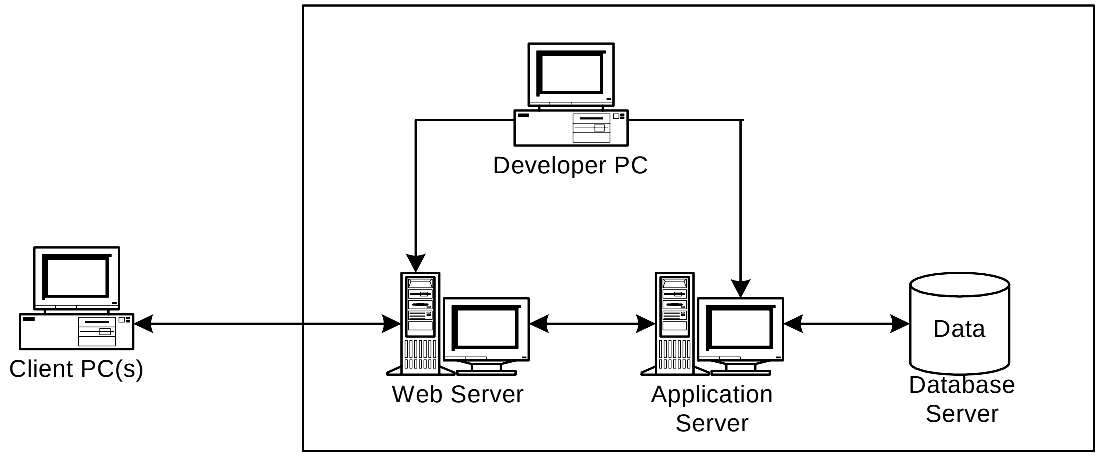
	PDF printing, and provides easy connectivity to Messaging Queues. Appeon Web application middle-tier can invoke Java Components, External DLL Functions, and C++ Components. Furthermore, it supports Web Services for the most flexibility integrating with J2EE and .NET.		
Database Server	Hosts the database for Web applications, providing the database connection to the application server.	None	Oracle, Sybase, Microsoft, or IBM.

The Developer PC will ultimately deploy your PowerBuilder application to the n-tier architecture outlined in Table 1-2.

Table 1-3: Developer PC Configuration

Tier	Purpose	Appeon Component	Third-party Software
Developer PC	Transforms PowerBuilder applications into Appeon Web applications, automatically.	Appeon Developer	PowerBuilder

1.4 System requirements



Client

- IE 6.0 SP1 for Microsoft Windows operating systems
- Optional: Adobe Acrobat Reader 6.0 for viewing PDF printed DataWindows and reports

Web Server

- Apache, Appeon Server, iPlanet, Microsoft IIS, and Netscape for Windows
- Microsoft Windows 2000 SP4 or NT 4.0 SP6a or XP SP1

Application Server

- Sybase EAServer 5.0 (Developer, Advanced or Enterprise Edition)

- Sybase PowerBuilder VM 8.0.4 Build 10656 or 9.0.1 Build 7171
- Sun® Java 2 JDK 1.3.1 patch level 06 or JDK 1.4 patch level 01
- Microsoft Windows 2000 SP4 or XP SP1

Database

- Microsoft SQL Server 2000 with Microsoft JDBC driver
- Oracle 8i or 9i with Oracle JDBC driver
- Sybase ASA 7.0.4 or ASA 8.0.2 or 9.0 with Sybase iAnywhere JDBC driver
- Sybase ASE 12.x with Sybase jConnect JDBC driver
- IBM DB2 UDB 8.1 with IBM JDBC driver

Developer PC

- Sybase PowerBuilder Enterprise Edition 8.0.4 Build 10656 or 9.0.1 Build 7171
- Microsoft Windows 2000 SP4 or NT SP6a or XP SP1
- Connectivity to Sybase EAServer 5.0 (Developer, Advanced, or Enterprise Edition) or local installation

2 Appeon Web Application Architecture

2.1 Standard technology

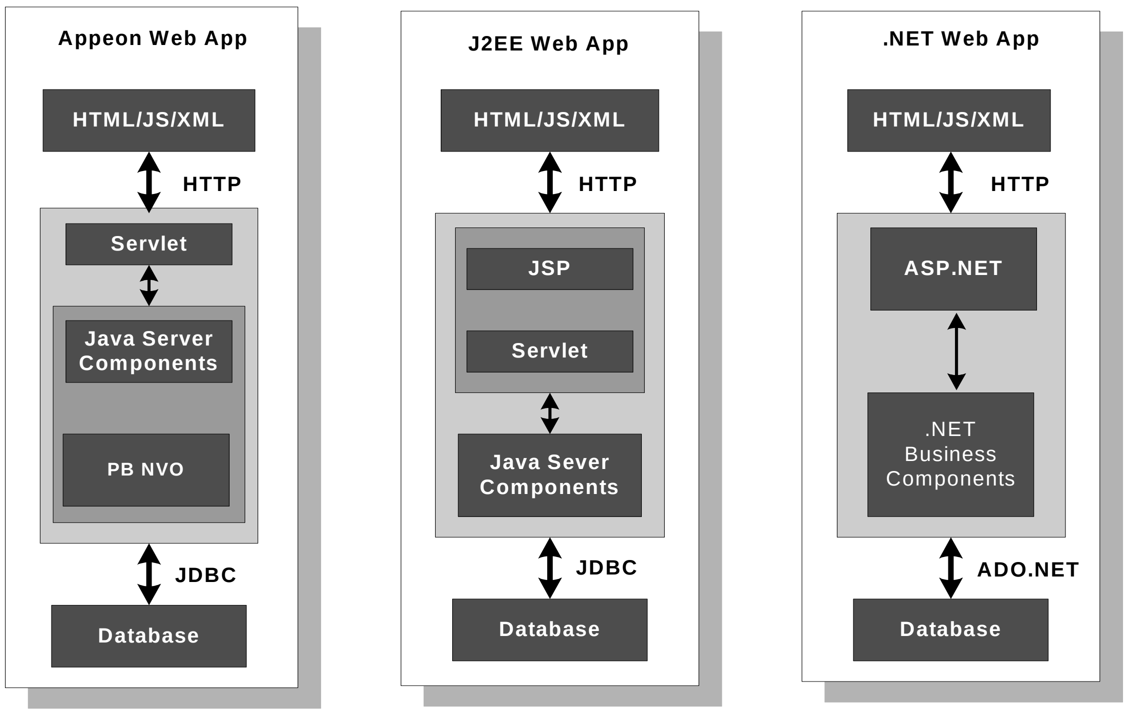
The Appeon architecture is implemented using industry standard technologies that are open and proven.

Appeon is a zero-client-installation solution. Appeon Web applications do not require the user to download browser plug-ins, Java Applets, ActiveX controls, Flash or any other client software. The user only downloads and displays/runs HTML, JavaScript, and XML.

The middle-tier of the Appeon architecture is largely J2EE-based. Only two of the many server components that make up the Appeon Server run-time framework are not J2EE CORBA server components. Two n-tier PB NVOs provide Image DataWindow generation functionality and DataWindow data connectivity. The only other non-Java code running on the middle-tier are any PowerBuilder business logic that the customer deployed to the Appeon Server as n-tier PB NVOs, if any.

Only standard communication protocols are used. The browser sends requests and data to the Web tier using HTTP/HTTPS. Within the middle tier, J2EE CORBA components communicate with PowerBuilder CORBA components using IIOP. The middle tier queries the database tier using SQL and JDBC. An ODBC back-end can be supported as well using a production-quality JDBC-ODBC bridge driver.

Appeon Architecture Comparison



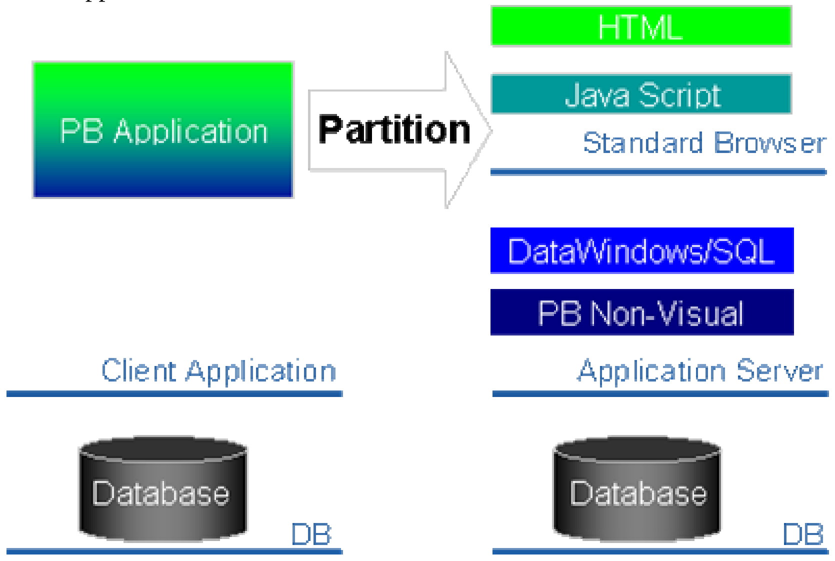
2.2 Browser-based n-tier architecture

Appeon Web applications deploy to the standard n-tier Web architecture – the presentation tier, middle tier, and data tier are logically separated.

- **Presentation Tier (Web Browser)** – The Web application presentation layer is implemented using only HTML and JavaScript. The HTML pages define the layout of the Web application windows. The

JavaScript files contain the JavaScript equivalent of the PowerScript or UI logic coded in the Window object, user object, menu object and any other visual objects. The Appeon Web application presentation layer runs in a standard Internet Explorer Web browser yet offers the same rich GUI as the source PowerBuilder application.

- **Middle Tier (Application Server)** – The middle tier, which hosts the business logic, is implemented with J2EE CORBA components and PowerBuilder CORBA components. The J2EE CORBA components execute the DataWindows and any Embedded SQL coded in the presentation layer of the PowerBuilder client/server application. The PowerBuilder CORBA components host the business logic of the application, if any has been coded as n-tier NVOs. The middle tier deploys to the J2EE-compliant Appeon Server, leveraging dozens of man-years of investments in DataWindows and other business logic.
- **Data Tier (Database)** – The database stores the raw data for the application and stored procedures, if any. The existing database from the PowerBuilder application can be simply re-used without modification so long as it is an Appeon-certified database.



When the developer clicks the "Deploy" icon of the Appeon Developer toolbar, the PowerBuilder client/server applications is first partitioned, then translated into Web languages, and lastly deployed to the n-tier architecture. The deployment process generates a set of Web files comprising HTML, XML and JavaScript. These files are deployed to the Web server. The DataWindow definitions (syntax) are uploaded to the Appeon Server. If the application contains n-tier NVOs, the user deploys these objects to Appeon Server using PowerBuilder.

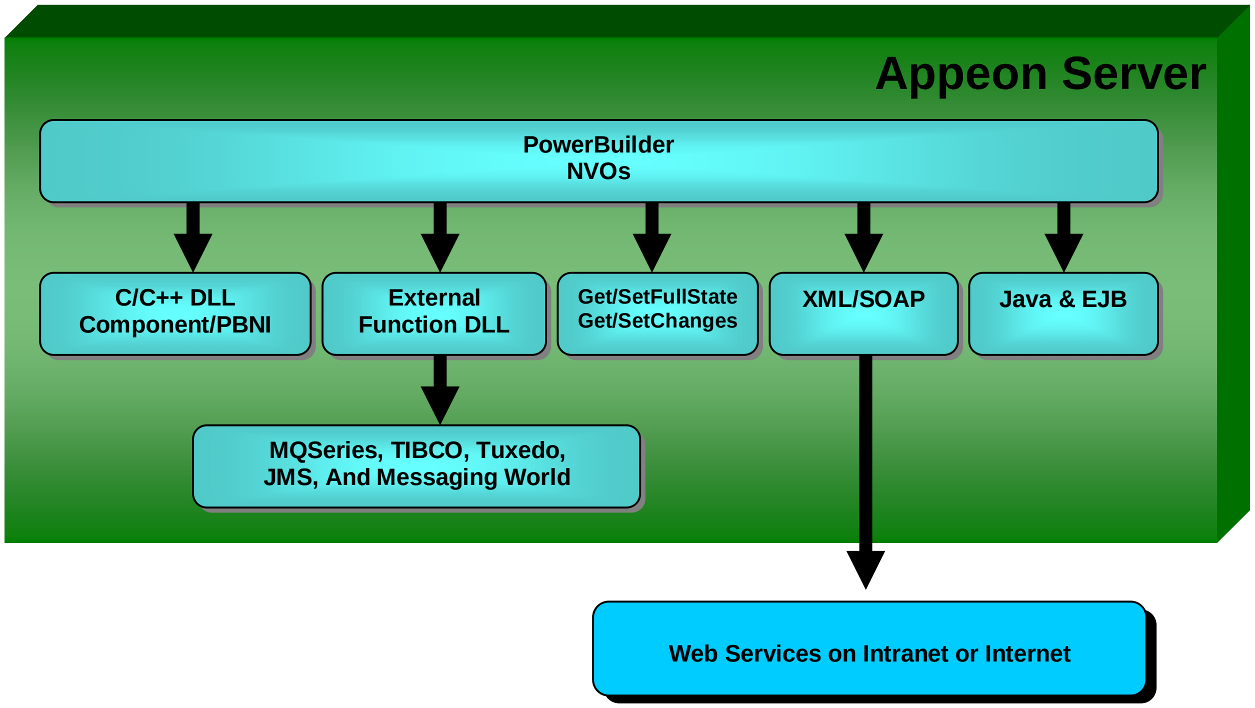
Table 2-1: Output of Web Deployment

Object Type	Web Deployment Output	Web File Size	% Cached
Entire Application	Index.htm entry page	.HTML: 1-25KB	100%
Window, Visual User Object	HTML file and JavaScript file	.HTML: 1-15KB .JS: 1-50KB	100%
DataWindow	DataWindow syntax uploaded to Appeon Server, which generates an XML file during run time.	.XML: 1-50KB	100%
Client-side NVO, Menu, Application Object	JavaScript File	.JS: 1-50KB	100%

2.3 Open and flexible J2EE/.NET integration

Appeon Web applications can be integrated with the following application types:

- .NET applications
- J2EE applications
- PowerBuilder applications
- Traditional Windows-based applications
- Other applications that support SOAP, IIOP, or PBNI protocols.


Appeon Server
PowerBuilder NVOs
C/C++ DLL Component/PBNI
External Function DLL
Get/SetFullState Get/SetChanges
XML/SOAP
Java & EJB
MQSeries, TIBCO, Tuxedo, JMS, And Messaging World
Web Services on Intranet or Internet

This flexible integration is made possible by Appeon's open middle tier that supports all leading industry standards for application integration.

- Appeon's middle tier supports SOAP and Web services. This relatively new but revolutionary standard enables application to be integrated with essentially any Web Services-enabled application over the Internet and without having any knowledge of the internal workings of the application or proprietary adapters. PowerBuilder NVO Components as well as Java Components and C++ Components on the Appeon Server can be easily exposed as Web Services such that other applications can readily access the business logic inside of them. Conversely, Appeon Web applications can also consume Web services.
- Appeon's middle tier also supports IIOP and CORBA components based on the Java, C/C++, and PowerBuilder programming languages. As such, Appeon's middle tier can directly invoke methods of EJBs (Enterprise Java Beans), COM/ActiveX components, DLLs (Dynamic Linked Libraries), and PB NVOs (PowerBuilder Non-Visual Objects) that are running on the application server locally. PowerBuilder NVOs can also easily access External DLL Functions of Windows applications to add in more integration possibilities.
- Appeon's middle tier also supports PBNI (PowerBuilder Native Interface), a set of C++ interfaces that enables C++ classes to harness the power of the PowerBuilder programming language and conversely enables C++ classes to be integrated into PowerBuilder/Appeon applications.

- Appeon's middle tier also supports passing DataWindow data and state between Server and Client including GetFullState/SetFullState/GetChanges/SetChanges for Composite, Crosstab, FreeForm, Graph, Grid, Group, Label, Nested, N-Up, and Tabular presentation styles. This allows Appeon Web applications to integrate with other visual and non-visual DataWindows external to the Appeon Web application.
- Appeon's middle tier also supports messaging through all standard messaging products including MQSeries, Tibco, Tuxedo, and JMS (Java messaging Server).

2.4 Strong Web security

Appeon supports the leading Web security standards and measures to ensure that all data transmissions are safe, secure, and authentic.

- First and foremost, Appeon Web applications are compatible with all corporate firewalls since Appeon communicates using HTTP over port 80 and only Web documents pass through the firewall (e.g. .HTML, .XML, .JS files).
- SSL encryption (HTTPS) up to 128-bits can be applied to all data transmissions to protect even the most sensitive data transmissions. This level of encryption is so secure that the USA government forbids exporting Web browser software with 128-bit encryption overseas.
- Digital certificates may be used to ensure that the Appeon Web application and all data transmissions are authentic. That is, the application and data transmissions are in fact from the specified party/server, and that the application and data transmissions have not been altered or corrupted in any way.

Appeon conforms to the strict Web browser security sandbox to ensure that the client computer system security cannot be compromised.

- The Appeon client-side utilizes only non-invasive Web technologies that cannot bypass the Web browser security sandbox.
- It is implemented using only HTML, JavaScript, and XML. There are no Java Applets, ActiveX controls, Flash, browser plug-ins, or other client software required other than the standard Microsoft Web browser.
- Thus Appeon Web applications cannot access the user hard drive, Windows registry, operating system, or anything outside of the Web browser. There is virtually no higher level of client security available for the Web.

Appeon's built-in multilevel application security ensures that the system cannot be accessed by unauthorized users, even when deployed to many different users over public networks (Internet).

- Most existing PowerBuilder application security measures are automatically replicated in the Appeon Web application. This includes features such as specifying privileges for accessing particular menus, windows, functionalities within windows, and even DataWindow data (columns).
- Appeon adds a second layer of application-level security on top of the existing PowerBuilder application security. Application level security will authenticate users based on logon credentials (e.g. username/password and IP address) before allowing the user to logon to the application. The user access can be managed using an LDAP server or Appeon's on built-in system.
- Session timeouts can be easily applied to all Appeon Web applications by simply configuring a setting in the AEM (Appeon Enterprise Manager). This feature helps safeguard the application from unauthorized access when authorized users have stepped away momentarily or forgot to logout from the system.

3 Appeon Web Lifecycle

3.1 Lifecycle of traditional JSP/ASP applications

The workflow of ASP/JSP applications is bound to the page metaphor, where the smallest unit of communication is a page. In other words, it is not possible for the client to just obtain one piece of data or execute one function independent of the ASP/JSP page. Any new information or processing must be performed by executing the ASP/JSP page on the server, generating an entirely new Web page that the user must download. Generally speaking, the majority of the data and processes of these ASP/JSP pages is redundant. In this respect, typical JSP/ASP Web applications are a step backward from the client/server world.

We can take a simple example of a master-detail DataWindow to illustrate the page metaphor. In this example, we will first load the DataWindow and retrieve data. Then, we will select a new master record (row in the master DataWindow) to obtain its details (displayed in the detail DataWindow).

To load the DataWindow initially, the lifecycle of the request would be as follow:

- The Web browser sends an HTTP request to the server for a JSP page that has master-detail DataWindows.
- The Servlet engine will execute the server-side Java code contained in the JSP page and retrieve data from the database for the master and detail DataWindows.
- The JSP page will be dynamically generated into an HTML page by the Servlet engine.
- The Web browser will download the HTML page from the server.
- The Web browser will render the HTML page, processing the document's markup language.

To load the details of a new master record, the lifecycle of the request would be as follows:

- The Web browser sends an HTTP request to the server for the same JSP page that has the master-detail DataWindows. The HTTP request contains a parameter - the ID for the new master record that has been selected.
- The Servlet engine will re-execute the server-side Java code contained in the JSP page and re-retrieve data from the database for the master and detail DataWindows. The execution of a good portion of this logic will be redundant but necessary to construct a new Web page and the data retrieval of the master DataWindow is redundant.
- The JSP page will be dynamically generated into an HTML page by the Servlet engine. The majority of the Web page has not changed except for the data of the detail DataWindow. Thus, all generation of HTML is redundant.
- The Web browser will re-download the HTML page from the server. This download is largely redundant. The exact amount of redundancy or waste can be measured by subtracting the file size of the data for the detail DataWindow from the total Web page file size.
- The Web browser will re-render the HTML page, re-processing the document's markup language. This results in the flash that is observed when users click on one master record and another.

It is possible to reduce the amount of redundancy by chopping up a single page into many small pages using frames. Essentially, you would put the master DataWindow into one page and the detail DataWindow into another page. Then you would create a third page to load the master DataWindow into one frame and the detail DataWindow into another frame. However, chopping up a page as described makes the application more difficult to develop and especially difficult to manage. That is, the project cycle and risk will increase. Furthermore, it is not feasible to chop up a Web page into infinite number of pieces. As such, there are limits on how much redundancy can be avoided using this method.

3.2 Lifecycle of Appeon HTML applications

The workflow of Appeon Web applications is based on the client/server metaphor, where the unit of communication is as granular as a function call or piece of data. Appeon Web applications are composed of a static set of HTML and JavaScript files that essentially become the Web client. The Web client will then execute logic at the client-side. If a piece of data is needed or some function on the server must be executed, the Appeon Web application will request that from the server through an HTTP-based RPC (remote procedure call) that returns results in XML. This is a very similar to Web services except that it is used for client-to-server communication instead of server-to-server communication. Since the Web client can be fully cached at the client-side (in the Temporary Internet Files folder), eventually, the only traffic between the client and the server is data and function calls.

We will reuse our simple example of a master-detail DataWindow to illustrate the Appeon lifecycle. In this example, we will first load the DataWindow and retrieve data. Then, we will select a new master record (row in the master DataWindow) to obtain its details (displayed in the detail DataWindow).

To load the DataWindow initially, the lifecycle of the request would be as follow:

- The Web browser sends an HTTP request to the server for an HTML page and a JavaScript file that corresponds to a PowerBuilder application window containing master-detail DataWindows.
- If the HTML page and JavaScript file do not exist in the Web browser cache (Temporary Internet Files folder), the Web browser will download them from the server. Otherwise, it will skip the download and obtain the files directly from the Web browser cache.
- The Web browser will render the HTML page and execute the logic of the JavaScript file.
- The Web browser will send an HTTP request (actually HTTP-based RPC) to the server to retrieve data from the database for the master and detail DataWindows.
- The server will generate two XML files containing the data for the two DataWindows.
- The Web browser will download the XML files from the server.
- The Web browser will parse the XML files and bind the data to the master and detail DataWindow controls rendered in the HTML page.

To load the details of a new master record, the lifecycle of the request would be as follows:

- The Web browser will send an HTTP request (actually HTTP-based RPC) to the server to retrieve data from the database for the detail DataWindow only. The RPC contains a parameter - the ID for the new master record that has been selected.
- The server will generate one XML files containing the data for the detail DataWindow only.
- The Web browser will download the XML file from the server.
- The Web browser will parse the XML file and bind the data to the detail DataWindow control already rendered in the HTML page.

Appeon distributes out the presentation layer processing to the Web browser whereas JSP/ASP applications require all processing to be done at the server. With Appeon, the server executes only business logic and data access logic, providing even greater partitioning of the application's logical tiers across the n-tier architecture. The processing power of the client is harnessed, reducing the server load. But most importantly, by having the presentation layer processed at the Web browser, the Web client becomes "smart" in that it can work at the granular level of data and function calls. HTML clients that are "dumb" do not know any better except to ask for another page that carries a significant cost.

3.3 Pros and cons of the Appeon "smart" presentation layer

Appeon's "smart" presentation layer has a number of benefits over the typical "dumb" ASP/JSP presentation layer:

- **Bandwidth saved.** Appeon reduces bandwidth consumption significantly. Virtually 100% of the files downloaded can be cached at the client-side except for data result sets. Furthermore, since Appeon does not generate any redundant content, downloads are only for what is absolutely necessary. This saves bandwidth. But more importantly, enables Appeon to deliver a very rich user experience without burdening bandwidth.

- **Server scalability improved.** Appeon boosts server scalability in two ways. First and foremost, Appeon harnesses the computing power of the client-side. As such, presentation layer processing can be moved from the server-side to the client. Second, the server is not burdened with generating redundant pages, which implies redundant execution of code on the server. As such, Appeon is arguably more scalable than even ASP and JSP Web applications.
- **Database scalability improved.** The database is the most critical tier in any application architecture in terms of scalability and performance. Appeon eliminates all redundant database retrievals, conserving precious database resources and boosting database scalability. There is no higher level of database efficiency that can be achieved without using special database caching and performance products, all of which Appeon is compatible with.
- **Better user experience.** Appeon delivers the rich PowerBuilder GUI and MDI interface on the Web using the standard HTML. Page refreshes become a thing of the past. The browser “back” and “forward” buttons go out in the thrash. The rich user interface of client/server was a huge step forward from dumb terminals and mainframes. With Appeon, there is no need to go backwards in order to reap the benefits of the Web and n-tier architecture.
- **Higher developer productivity.** There is no need to complicate your application development with frames and give up the productivity of 4GL RAD. In PowerBuilder, developers can build complex Windows-style user interfaces using an event-driven programming model and readily integrate them with DataWindows and data sources. Just recently has VisualStudio.NET and a number of J2EE frameworks popped up that provide some level of component-based development, which is just one of the many features that makes PowerBuilder highly productive.

Of course, there is a tradeoff to having such a “smart” client running at the Web browser, client-side runtime performance. If the client-side is made too “heavy” with a large number of DataWindow controls, for example 25 DataWindows, 11 DropDownDataWindows, and 5 DataStores, it may take some time to open such a PowerBuilder window on the Web. It is really easy to get carried away with DataWindows when you are coding in PowerBuilder for the client/server architecture. However, if you compare this to typical JSP Web applications with Sybase Web DataWindow technology for the Web architecture, you would generally not see one JSP page crammed with 40+ DataWindows in the first place. You would not see thousands of rows of data being displayed in a single page. So all in all, if some basic discretion is exercised and Web best practices are employed, it is possible to have all the benefits of a “smart” HTML client and good client-side runtime performance without any compromise.

You can read more about Appeon’s client-side runtime performance and performance tuning in the Appeon product documentation titled “Appeon Performance Tuning Guide” that ships with the Appeon for PowerBuilder product.

4 Conclusion

The Appeon architecture delivers all the benefits of the Web and client/server to standard Microsoft Web browsers, based on open standards and adopting a true n-tier Web architecture. Microsoft and the J2EE community are making strides to introduce a richer user interface for standard Web browsers. Microsoft ASP.NET's new Web Forms is one example and JSF (JavaServer Faces) is another. However if you compare what Appeon has accomplished with HTML, JavaScript, XML, and HTTP(S), along with the power of DataWindows on the Web, you will see Appeon's rich user interface is generations ahead of everyone else. All your new Web development projects and PowerBuilder client/server migration projects are built with the unparalleled 4GL productivity of PowerBuilder. When you are ready to go mobile, your Appeon Web applications can be seamlessly extended to PocketPC using Pocket PowerBuilder without the expense and risk of an entirely new project.

Appeon Corporation
1/F, Shell Industrial Building
12 Lee Chung Street
Chai Wan District, Hong Kong
www.appeon.net