

New Features for PowerBuilder 11.0

The features described in this document are available in beta 2. For more information about using these features, see the New Features section in the online Help. For a list of known issues, see the PB110readme.txt file.

New Target Types

PowerBuilder 11.0 introduces new target types. The target types include:

- Application (the legacy PowerScript application target)
- .NET Windows Forms Application
- .NET Web Forms Application
- .NET Web Service
- .NET Assembly
- EAServer Component
- Application Server Component
- Java Server Pages (same as in earlier PowerBuilder releases)

Each target type requires the creation of a project that you use to deploy, run, or debug the target application or component. For component targets, you can indicate the application you want to use to test your component in the development environment.

.NET Web Forms Deployment

With PowerBuilder 11, you can deploy PowerBuilder applications that will run in a browser as ASP.NET 2.0 applications. To do that, create a .NET Web Forms project using the .NET Web Forms Application Wizard and deploy the application from the .NET Web Forms Project painter. You can test the Web application by right-clicking on the project in the System Tree and selecting the Run Project menu item in the context menu. Your end users will access the deployed application through a browser with a URL that you provide. If your PowerBuilder application is an MDI application, the MDI sheets are displayed as tab pages in the deployed Web application when multiple sheets are opened.

Most PowerBuilder features, including embedded SQL, file operations, DataWindow printing, and calling external functions, are supported in Web Forms applications. Features that are not appropriate for Web deployment, such as drag-and-drop, are not supported. When deploying an application, PowerBuilder reports all unsupported features used by the application in the Output window. The Web Forms deployment feature is suitable for applications of moderate complexity that use query, data transaction, and reporting features. Typically, only minor adjustments to PowerScript code are necessary before you deploy PowerBuilder applications as .NET Web Forms applications.

This feature enables you to easily and quickly deploy PowerBuilder applications to the Web without incurring a steep learning curve for building and deploying .NET Web Forms applications. You can leverage existing skills and experience in PowerBuilder to deploy applications to ASP.NET servers.

.NET Windows Forms Deployment

With PowerBuilder 11, you can deploy PowerBuilder applications as .NET Windows Forms applications. To do that, create a .NET Windows Forms project using the .NET Windows Forms Application Wizard and deploy the application with the .NET Windows Forms Project painter. You can then test the Windows Forms application by right-clicking on the project in the System Tree and selecting the Run Project context menu item, or by selecting Run from the toolbar of the Windows Forms Project painter.

Most PowerBuilder features, including drag-and-drop, registry, and clipboard functions are supported in Windows Forms applications. In most cases, these features behave the same as they do in client-server PowerBuilder applications, but may have a slightly different appearance. Please refer to the online Help for further details.

The Windows Forms deployment feature is an extension of the traditional PowerBuilder Win32 application in the .NET environment. You can create .NET Windows Forms applications in PowerScript syntax without incurring the learning curve often necessary for application deployment to the .NET platform. With the .NET interoperability features of PowerBuilder 11, you can leverage functionality provided by the .NET Framework and third-party tool vendors.

Smart Client Deployment

In the past few years, Web-based applications have dominated the application deployment market. One of the key reasons to choose Web-based applications is the ease of deployment. The PowerBuilder 11 smart client deployment feature is designed to leverage this benefit and simplify application deployment. Smart client applications bring together the best of both worlds: a combination of the richness your end users are used to with client-server based applications combined with the ease of deployment of Web applications.

PowerBuilder 11 has a new component called the intelligent updater that you can use to make .NET Windows Forms applications self-updatable quickly and easily. Enable smart client deployment in the .NET Windows Forms Application Wizard to take advantage of this feature. For further details, please refer to the online Help.

Deploy Nonvisual Objects as .NET classes in .NET Assemblies

Nonvisual objects can now be deployed as .NET classes in .NET Assemblies. During deployment, the namespace can be specified and the class and function names can be modified in the deployed version. Standard data types such as int and char can be exposed as .NET nullable types.

.NET Debugger

With PowerBuilder 11, you can debug PowerBuilder .NET applications in the PowerBuilder debugger. To do that, you must first deploy the .NET target. You can invoke the debugger by clicking the "Debug" icon in the toolbar, or by right-clicking on the project in the System Tree and selecting the Debug menu item.

The .NET Debugger enables you to easily debug PowerBuilder .NET applications, since it has almost the same operations as the PowerBuilder native debugger. Most PowerBuilder debugging features, including expression evaluation and conditional

breakpoints, are supported in .NET applications. The “Objects in Memory” view and variable breakpoints are not supported due to limitations of the .NET platform. Exception handling is enhanced in the PowerBuilder .NET Debugger. It also provides some unique features like attaching to or detaching from a current .NET process.

You can leverage existing PowerBuilder debugging skills and experience in debugging .NET applications and components.

Nonvisual Objects for .Net Web Service Deployment

Web services are ideal for cross-platform communication in heterogeneous environments because of their use of open standards such as XML and the Simple Object Access Protocol (SOAP). The ease of deployment and maintenance makes Web services a very attractive approach.

PowerBuilder .NET Web Services components are built on top of the Microsoft ASP.NET Web Service framework. When you deploy a .NET Web Services target, the PowerBuilder .NET Web Services generator creates .asmx files and the .disco file for selected PowerBuilder nonvisual objects.

The .NET Web Service project wizard can help PowerBuilder developers create a .NET Web service project quickly and easily. It guides the developer through a series of steps, collecting the required information for deploying the project. You can use the Project painter to view and edit information entered in the wizard. After you deploy the Web service, you can run and debug Web service methods from a test application that you assign to the .NET Web Service project in the Project painter.

Conditional Compilation

Conditional compilation is useful for things like machine dependencies, platform dependencies, debugging, and for setting certain options at compile time. PowerBuilder 11.0 provides this feature to allow developers to differentiate among target types when writing code at development time. PowerBuilder provides five preprocessor symbols for different target types and a “DEBUG” symbol for debugging purposes. With conditional compilation, users can share the common code and write special code for different target deployment environments.

.NET Language Interoperation

The .Net framework and other associated third party managed libraries provide a very rich resource. PowerBuilder users can use these libraries to extend the functionality of PowerBuilder and save development time.

.NET language interoperability makes it possible to consume .NET classes and methods in PowerBuilder applications. With .NET language interoperability, you can use PowerBuilder syntax to create .NET classes, call .NET methods, and access .NET properties. You can make use of .NET collection classes, such as Hashtable and Set, and you can also make use of powerful .NET communication classes and other .NET services.

Smart Client Data Access using MobiLink Synchronization

Sybase MobiLink offers data synchronization using scripts and roles that allow “occasionally connected” users to work offline with data specific to their needs. These users can easily synchronize their data with the consolidated database whenever they are connected.

PowerBuilder 11 allows developers to create applications that can make use of this technology with a built-in Standard Class.

TreeView DataWindow for Web Forms

The TreeView DataWindow for Web Forms is useful to display hierarchical data on the Web. It can freely expand or collapse information to show or hide specified details. This feature will benefit PowerBuilder developers who want to show hierarchical data on the Web.

Web Service DataWindow

The Web Service DataWindow allows the DataWindow to use a Web Service method for data retrieval. The Web Service method used for retrieval must return the data in one of the following forms:

- a simple type
- an array of simple types
- a structure made up of simple types
- an array of structures

Microsoft SQL Server 2005

PowerBuilder 11 includes a new interface (SNC) to support SQL Server 2005. This interface is similar to the Microsoft OLE DB interface, but it uses Microsoft’s new OLE DB driver dedicated to SQL Server 2005. With this interface, PowerBuilder applications can leverage the new features provided by Microsoft SQL Server 2005.

Contemporary Menus and Toolbars

The PowerBuilder development environment has a fresh look with menus and toolbars that have a more modern appearance. The “contemporary style” toolbar introduced for PowerBuilder 10.5 applications is now featured in the development environment, with gradient background shading and a style familiar to developers who work with Microsoft Office 2003 and Visual Studio 2005.

Current Target Highlighted in the System Tree

The current target is now indicated in the system tree, and menu items have been added for building and deploying the current target. Keyboard shortcuts can be defined for these menu items.

Project Painters are Greatly Enhanced

The Project painters in PowerBuilder are now consistent in their editing actions and usability. You edit project properties without modal dialogs. For many of the projects, you can indicate the application you want to use to test your project.

Imported .NET Assemblies in the System Tree

The System Tree now displays imported .NET assemblies that you associate with your target. The System Tree contains the types, methods, and properties for the assemblies that you can drag and drop onto Script views—just like any other item in the System Tree.

You can add, remove, or reorder imported .NET assemblies on the new .NET tab on the target Properties dialog box for any .NET target type.

Functions and Events with Code in the System Tree

The System Tree now groups functions and events with actual implementations and has a new icon that indicates when there is code associated with functions and events. This feature makes it easier to use the System Tree to find code that you want to edit. Double-clicking an event or function opens it in the Script view.

Many Dialog Boxes are Resizable

The “File > New” and target Properties dialog boxes and many other dialog boxes are resizable.

PowerBuilder Development Environment Licensed with SySAM

The PowerBuilder and InfoMaker development environments are now using the Sybase standard “SySAM” licensing tool. This flexible model is the same as that used by Adaptive Server Enterprise 15, EAServer 6, and PowerDesigner 12. Details of the current license are displayed in the “About” dialog box.

DataWindow Retrieval Values can be Saved

In the PowerBuilder development environment, the values that you use for DataWindow retrieval can be saved for later use. This saves the step of entering values each time you need to retrieve the DataWindow at design time.

AutoScript has Option to Display Return Values

AutoScript has an option to display the return values for methods.

SCC Warning Dialog Box Optional

When opening an object that is under source control, but not checked out, the dialog box informing you that the object is not checked out is now optional.

Building with Painters Open

Building a target with painters open is now possible. Painters which are “dirty” still prevent the build from proceeding, but the overall build process is now much smoother when no changes have been made to the objects in an open painter.

