New Features PowerBuilder® 11.2

Document ID: DC00357-01-1120-01

Last revised: March 2008

AJAX update functionality for Web Forms applications

PowerBuilder® 11.2 introduces AJAX (Asynchronous JavaScript and XML) update functionality for Web Forms applications. With ASP.NET AJAX, Web Forms pages are updated by refreshing individual regions of each page asynchronously.

Asynchronous postbacks enhance application performance by minimizing the content requested from the Web server and rendered again in the client browser. The rest of the Web page remains unchanged, therefore data traffic and page flickering are significantly reduced.

The AJAX enhancement for Web Forms applications does not require any changes in the PowerBuilder IDE. To use the AJAX feature, you do not need to learn anything new or alter your PowerScript® code.

However, PowerBuilder cannot deploy a Web Forms application unless AJAX is installed on the Web server. You can download and install the Microsoft ASP.NET AJAX Extensions version 1.0 on both the development and deployment computers from the ASP.NET Web site at http://www.asp.net/ajax/downloads/archive.

For information on how Web Forms applications work with AJAX extensions, see the chapter on "Overview and Configuration of .NET Targets" in the *Deploying Applications and Components to .NET* book.

Using synchronous update functionality
If you must use synchronous update functionality, you can change the PBPostbackType global property to "Synchronous" in the *Web.config* file that PowerBuilder creates when you deploy your Web Forms application. However, support for synchronous update functionality will be removed in future releases of PowerBuilder and you cannot modify the PBPostbackType property in the PowerBuilder IDE.

For information on global properties, see the chapter on "Properties for .NET Web Forms" in the *Deploying Application and Components to .NET* book. For information on modifying the *Web.config* file, see the section on "Viewing and modifying global properties in IIS Manager" in the same book.

AJAX waiting message properties

In PowerBuilder 11.2, six new global properties allow you to set text and formatting properties for the messages that your applications display to end users while they wait for new content to be rendered in their browsers. The messages display only if you use the default asynchronous AJAX update functionality for your Web Forms applications.

You can set the global properties at design time on the Configuration tab of the Project painter or after deployment in the generated *Web.Config* file for your application. Support for message text localization is currently limited, but will be enhanced in future releases of PowerBuilder.

The new global properties are listed and described in the following table:

Property	Default value	Description
PBAjaxWaitingMessage	Loading ... ... Please wait...	Status message text displayed to the user during AJAX request processing. To disable the status message set the value to an empty string ("").
PBAjaxWaitingMessage FontFamily	Tahoma	Font family for the text of the AJAX waiting message.
PBAjaxWaitingMessage FontSize	10	Font size of the text of the AJAX waiting message.
PBAjaxWaitingMessage BoxPosition	Center	Position of the AJAX waiting message box. Valid values are: Center, TopLeft, TopRight, BottomLeft, and BottomRight.

Property	Default value	Description
PBAjaxWaitingMessage BoxHeight	—	Height in pixels of the AJAX waiting message box. If no value is specified (default), PowerBuilder calculates the best fit for the message box text.
PBAjaxWaitingMessage BoxWidth	—	Width in pixels of the AJAX waiting message box. If no value is specified (default), PowerBuilder calculates the best fit for the message box text.

Telerik RadControls support

PowerBuilder 11.2 uses Telerik RadControls for menus, toolbars, and other controls in Web Forms applications by default. Although not recommended, you can use IE Web Controls instead of the RadControls, but you must change the PBWebControlSource global property for your application and install the IE Web Controls on the server.

Telerik RadControls provide enhanced functionality for Web Forms toolbars and menus, DatePicker and MonthCalendar controls, and TreeView controls. There are minor display differences for TreeView controls, depending on your PBWebControlSource selection.

For information on the effect of the PBWebControlSource selection on the HasButtons and Indent TreeView control properties, see the chapter on “Modified and Unsupported Features in Web Forms Projects” in the *Deploying Applications and Components to .NET* book.

If you must use IE Web Controls, you need to download IE Web Controls from the Microsoft Web site at http://www.asp.net/IEWebControls/Download.aspx and install the controls on the Web server. After you install the controls, you must set the PBWebControlSource selection to “IE”, either on the Configuration tab of the Project painter before you deploy your application, or in the *Web.config* file after you deploy your application.

For information on installing IE Web Controls, see “Setting up IE Web Controls on the server” in the *Deploying Applications and Components to .NET* book.

Building .NET clients for EAServer

In PowerBuilder 11.2, you can build a .NET client application that invokes methods of Enterprise JavaBeans (EJB) components or PowerBuilder EAServer components running in EAServer 6.1 or later. This capability is based on the .NET client ORB library introduced in EAServer 6.1.

Installation requirement
When you install EAServer, you must install the .NET support option.

You can use either the Connection object or the JaguarORB object to connect to the component in EAServer, and you can connect from PowerBuilder .NET Windows Forms and Web Forms applications and from PowerBuilder .NET assemblies and Web services.

For information on the differences in behavior of Connection objects and JaguarOrb objects when you connect to EAServer from a .NET client, see the “Building .NET clients for EAServer” chapter in the *Deploying Applications and Components to .NET* book. This chapter also provides a table of supported CORBA datatypes and describes how to connect to EAServer using an SSL connection.

For detailed information on building an EAServer client, see the “Building an EAServer Client” chapter in *Application Techniques*.

Changing application pools for Web Forms in IIS 7

Virtual directories in IIS 7 are hosted in an application pool. An application pool is the host process for one or more Web applications. When you deploy a PowerBuilder Web Forms application to IIS 7 in PowerBuilder 11.2, the application is deployed to a PowerBuilder-specific application pool named PBAppPool. On 64-bit Vista, the PBAppPool pool is configured to run 32-bit applications.

To avoid compatibility issues with some features, such as the TreeView control, Web Forms applications deployed from PowerBuilder must run in an application pool that uses the classic managed pipeline mode, where ASP.NET runs as an ISAPI extension. The PBAppPool application pool uses the integrated managed pipeline mode by default, but you should change it to use the classic mode if you use TreeView controls.

❖ **To change the PBAppPool managed pipeline mode to classic:**

1. In IIS Manager, select Application Pools.
2. In the list of Application Pools, double-click PBAppPool.
3. Set Managed Pipeline Mode to Classic and click OK.

Using a certificate store for smart client applications

In PowerBuilder 11.2, you can select a digital certificate from a certificate store to sign your smart client application manifests. Previously you could only select a certificate from a file browser. You make the selection on the Publish page in the Project painter for a .NET Windows Forms target when Publish as a smart client application is selected on the General page. You can also select a password protected certificate.

A digital certificate is a file that contains a cryptographic public/private key pair, along with metadata describing the publisher to whom the certificate was issued and the agency that issued the certificate.

Digital certificates are a core component of the Microsoft Authenticode authentication and security system. Authenticode is a standard part of the Windows operating system. To be compatible with the .NET Framework security model, all PowerBuilder .NET applications must be signed with a digital certificate, regardless of whether they participate in Trusted Application Deployment.

Use the Select from Store or Select from File buttons to select a certificate from a certificate store or from your file system. If the certificate requires a password, a dialog box displays so that you can enter it. When you select a valid certificate, detailed information displays in the Project painter.

If you do not specify a certificate, PowerBuilder generates a test certificate for you automatically, but you should not deploy an application with a test certificate when you are ready to publish a production application.

For more information about Trusted Application Deployment, see the Microsoft Web site at http://msdn2.microsoft.com/en-us/library/01daf08f.aspx.

Usability and user interface enhancements

PowerBuilder 11.2 includes the following user interface enhancements:

- Target-relative paths and shared projects
- Library tab and wizard in the New dialog box
- Library list and .NET Assemblies context menu items on targets in the System Tree
- Publish context menu item on smart client projects in the System Tree
- Files opened in File editor added to most recently used objects list
- File editor Open dialog box lists additional file types
- Current database connection displays in the PowerBuilder title bar
- UseEllipsis property available in edit styles in the Database painter

Target-relative paths and shared projects

All paths used in projects are stored as target-relative paths, if possible. If you later move the application to a different location in the file system, or another user copies or checks out the application, the paths are adjusted relative to the new target location.

For example, suppose user A has an application target stored in the following directory structure, where *pbl_1.pbl* contains the application object:

```
C:\target1\target1.pbt
C:\target1\pbls\pbl_1.pbl
C:\target1\pbls\pbl_2.pbl
C:\target1\res\target1.pbr
C:\target1\out\target1.exe
```

When user B copies the application to the following directory structure, no changes need to be made in the Project painter, because the paths reflect the new directory structure:

```
D:\PB\My Targets\Target 1\target1.pbt
D:\PB\My Targets\Target 1\pbls\pbl_1.pbl
D:\PB\My Targets\Target 1\pbls\pbl_2.pbl
D:\PB\My Targets\Target 1\res\target1.pbr
D:\PB\My Targets\Target 1\out\target1.exe
```

A project that was created in an earlier version of PowerBuilder using hard-coded paths must be opened and resaved before the files it references are modified with target-relative paths.

However, if a path is not on the drive where the target is stored, then the path is stored as an absolute path. For example, the path to image files stored on a shared network directory such as *J:\res\images\common* is stored as an absolute path in the project file.

References to files outside the target path
If a project references a PBL or another file on a local drive that is outside the path of the target, make sure that the PBL or file is copied to the new target location and that it is referenced correctly in the project.

Library tab and wizard in the New dialog box

When you create a new target, PowerBuilder creates a new library automatically. If you need additional libraries, you no longer need to open the Library painter to create them. The New dialog box has a new Library tab with a Library icon that opens a wizard. In the wizard, you provide a name, location, and optional description for the library. When you click Finish, the new library is added to the target's library list.

If you open the New dialog box from the menu bar or PowerBar, the new library is added to the library list of the current target (the target in bold in the System Tree). If you select New from the pop-up menu of a specific target, the library is added to the library list for that target.

Library list and .NET Assemblies context menu items on targets in the System Tree

The pop-up menu for targets in the System Tree now contains a Library List item to make it easier to access the Library List page in the target's Properties dialog box. .NET targets also have a .NET Assemblies pop-up menu item that opens the .NET Assemblies page in the target's Properties dialog box.

You still need to close any open painters in the target to make changes to the library or .NET assemblies lists.

Publish context menu item on smart client projects in the System Tree

The pop-up menu for a .NET Windows Forms project that has the "Publish as smart client application" check box selected now contains a Publish item. You can deploy, run, debug, and publish the application from the System Tree, without opening the Project painter, if you do not need to change any settings in the painter.

Files opened in File editor added to most recently used objects list

The File>Recent Objects menu item lists PowerBuilder objects that were opened recently in painters or in the Source editor. In PowerBuilder 11.2, it also lists files opened in the File editor.

File editor Open dialog box lists additional file types

In PowerBuilder 11.2, you can select JavaScript or HTML from the Files of Type drop-down list in the File Open dialog box for the File editor.

Current database connection displays in the PowerBuilder title bar

If PowerBuilder is connected to a database, the three-letter abbreviation for the database interface followed by the name of the database profile displays in PowerBuilder's main title bar. If you are working with a DataWindow®, this visual cue makes it easier to check that you are using the right connection.

For example, if you open the PowerBuilder Code Examples workspace and connect to the EAS Demo database, the title bar displays "pbexamples - ODB [EAS Demo DB V110] - PowerBuilder."

UseEllipsis property available in edit styles in the Database painter

The UseEllipsis DataWindow object property was added in PowerBuilder 11.0. The property displays an ellipsis at the end of character data that is too long for a column with the Edit or EditMask edit style. In PowerBuilder 11.2, the property is available on the General page of the Object Details view in the Database painter when you select or create an Edit or EditMask edit style in the Extended Attributes view.

Database connectivity enhancements

PowerBuilder 11.2 includes the following database connectivity enhancements:

- DisableBind DBParm supported by ASE and SYC database interfaces
- Support for Oracle 10.2 NCHAR literal replacement

DisableBind DBParm supported by ASE and SYC database interfaces

For DBMSs that support bind variables, PowerBuilder can bind input parameters to a compiled SQL statement. The DisableBind database parameter allows you to specify whether you want to disable this binding. When DisableBind is set to 1 to disable binding, PowerBuilder replaces the input variable with the value entered by the application user or specified in code.

In PowerBuilder 11.2, the ASE and SYC native database interfaces for Adaptive Server® Enterprise support the DisableBind database parameter. The default value is 1. For more information, see the description of DisableBind in the online Help.

Support for Oracle 10.2 NCHAR literal replacement

By default, in a SQL statement, the text of any literal is encoded in the same character set as the rest of the statement. The character set on the client is determined by the client character set defined in NLS_LANG. When the statement is executed, the character set on the client is converted to the character set on the database server.

Data in string literals is lost in the conversion if the character set on the database server does not contain the characters used on the client. NChar string literals are most affected by this issue because they are designed to be independent of the character set on the database server.

To avoid this data loss, set the NCharLiteral database parameter to ‘Yes”. This setting causes the Oracle client to encode all literals prefixed with N in the statement on the client with an internal format. The database server decodes the literals to Unicode when the statement is executed.

For example, when NCharLiteral is set to “Yes”, the string “some unicode data” in the following SQL statement is transferred from the client to the server with no data loss:

```
insert into table1 (id, ncharcol) values(1, N'some
unicode data')
```

Oracle 10.2 or higher required
The NCharLiteral database parameter requires Oracle 10.2 or higher on both the client and the database server.

For more information, see the description of NCharLiteral in the online Help.

Enabling the DEBUG condition in ORCA and OrcaScript

PowerBuilder 11.2 includes a new property and method that allow you to programmatically compile standard PowerBuilder applications using or excluding script contained in conditional compilation blocks set with the DEBUG preprocessor symbol. The ORCA tool and OrcaScript commands do not use Project objects to compile applications of standard client-server PowerBuilder targets, even though the Project object is currently where you set inclusion or exclusion of the DEBUG condition while compiling from the IDE.

Windows Forms applications
ORCA and OrcaScript use Project objects to compile Windows Forms applications. For these applications, the Enable DEBUG Condition check box selection in a Project object determines whether or not the script in DEBUG conditional compilation directives is compiled—even when you compile code using ORCA or OrcaScript.

For information on the DEBUG conditional compilation directive, see "Using the DEBUG preprocessor symbol" in the PowerBuilder *User's Guide*.

ORCA methods

In ORCA 11.2, you can enable or disable the DEBUG condition for an entire ORCA session using the new boolean bDebug property of the PBORCA_CONFIG_SESSION structure. Otherwise, you can call the PBORCA_SetDebug method whenever you want to enable or disable the DEBUG condition during an ORCA session. This allows you to enable the DEBUG condition for some of the objects in a target and to disable it for other objects. The Pcode for an object stored in a target PBL reflects the DEBUG conditional setting in use when the object was last compiled or regenerated.

IDE regeneration
All objects compiled in the PowerBuilder IDE are compiled with the DEBUG condition enabled. When a PowerBuilder developer saves changes to an object or regenerates it in the IDE, the Pcode stored for the object in the target PBL automatically includes code from any DEBUG conditional directive scripted for the object.

For more information on the bDebug property and the PBORCA_SetDebug method, see the *ORCA Guide* PDF that the setup program installs in the *PowerBuilder 11.0\SDK\ORCA* directory.

OrcaScript method

The set debug OrcaScript command invokes the ORCA PBORCA_SetDebug method and can be called any time after the start session command. The value you set affects all objects used by subsequent regenerate and build application commands. It also affects all objects retrieved with scc refresh target and scc get latest version commands.

The build application full command in the following example recompiles all of the objects in the application PBL with the DEBUG condition disabled, and the *buildapp_p.exe* application created by the build executable command behaves exactly like a production application (without any debug code).

```
start session
set debug false
set liblist "testdebug\buildapp.pbl"
set application "testdebug\buildapp.pbl" "testdebug"
build application full
build executable "destination_1\buildapp_p.exe" "icon\icon9.ico" "" "N"
end session
```

Setting the debug value only affects objects that are compiled or regenerated after the set debug command is issued. The following example copies the PBL generated from the previous example after it was compiled with the debug condition disabled. In this example, even though set debug true is called before it builds the *debug_copy.exe* executable, the code in DEBUG conditional compilation blocks is not enabled because none of the commands that follow the set debug call invoke the PowerScript compiler.

```
start session
set debug TRUE
file copy "testdebug\buildapp.pbl" "testdebug\copy.pbl" clobber alwaysset
liblist "testdebug\copy.pbl"
set application "testdebug\copy.pbl" "testdebug"
build executable "destination_1\debug_copy.exe" "icon\icon9.ico" "" "N"
end session
```

However, if you add a build application command or a regenerate command after the set debug command in the previous example, the script inside DEBUG conditional compilation blocks will be enabled.