New Features PowerBuilder 11.0

Document ID: DC00357-01-1100-01

Last revised: May 2007

.NET deployment features

PowerBuilder 11 introduces the ability to build applications and components in PowerBuilder and deploy them to the .NET Framework version 2.0. New wizards let you deploy your PowerBuilder applications as .NET Web Forms or as Windows Forms applications with optional intelligent update (smart client) capability. You can also deploy custom class user objects as .NET Web services and assemblies.

- .NET Web Forms deployment
- .NET Windows Forms and smart client deployment
- Deploying nonvisual objects as .NET classes in .NET assemblies
- Deploying nonvisual objects as .Net Web services
- .NET debugger
- Conditional compilation
- .NET language interop
- .NET assembly import

For more information about all .NET features, see *Deploying Applications and Components to .NET*.

.NET Web Forms deployment

Web Forms applications have several advantages over traditional client-server and Windows Forms applications. Web Forms applications do not require client-side installation, are easy to upgrade, have no distribution costs, and offer broad-based user access. Any user with a Web browser and an online connection can run Web Forms applications. Simple inquiry, browsing, or reporting applications are suitable candidates for Web Forms deployment.

Use the.NET Web Forms Application target wizard to create a Web Forms target and deploy the application from the .NET Web Forms Project painter. After you deploy, you can test the Web application by right-clicking on the project in the System Tree and selecting the Run menu item in the context menu. Your end users will access the deployed application through a browser with a URL that you provide.

For more information, see “Web Forms Targets” in *Deploying Applications and Components to .NET*.

.NET Windows Forms and smart client deployment

Windows Forms applications with the smart client feature combine the reach of the Web with the power of local computing hardware. They provide a rich user experience, with a response time as quick as the response times of equivalent client-server applications. The smart client feature simplifies application deployment and updates, and can take advantage of Sybase’s MobiLink technology to provide occasionally connected capability. Applications that require significant data entry, retrieve large amounts of data (for example, more than 3 MB per request), or have a complex user interface are suitable candidates for Windows Forms deployment.

Use the .NET Windows Forms Application target wizard to create a Windows Forms target and deploy the application with the .NET Windows Forms Project painter. Enable intelligent deployment and update by selecting the smart client check box in the wizard or the painter. You can test the Windows Forms application by right-clicking on the project in the System Tree and selecting the Run menu item in the context menu. Click the Publish button in the Project painter to publish the application to a Web or FTP site or to the network file system so that your users can have easy access to the latest version.

For more information, see "Windows Forms Targets" in *Deploying Applications and Components to .NET*.

Deploying nonvisual objects as .NET classes in .NET assemblies

You can deploy nonvisual objects as .NET classes in .NET assemblies using the .NET Assembly target wizard and Project painter. Datatypes are mapped automatically from PowerScript to C#.

When you deploy a .NET Assembly project, PowerBuilder creates an assembly DLL fron the nonvisual user objects you selected in the wizard or Project painter. If you also listed a setup file name, PowerBuilder creates an MSI file that includes the assembly DLL, PowerBuilder system libraries for .NET, and any resource files you listed in the wizard or project painter.

For more information, see ".NET Assembly and Web Service Targets" in *Deploying Applications and Components to .NET*.

Deploying nonvisual objects as .Net Web services

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).

PowerBuilder .NET Web service components are built on top of the Microsoft .NET Framework. When you deploy a .NET Web Service target to IIS, PowerBuilder creates *.asmx* files and the *.disco* file for the PowerBuilder nonvisual objects you select in an application directory on the server.

The .NET Web Service wizard guides you through a series of steps, collecting the information needed to deploy the project. 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.

For more information, see ".NET Assembly and Web Service Targets" in *Deploying Applications and Components to .NET*.

.NET debugger

When you have deployed a .NET target, you can debug it in the PowerBuilder debugger. Invoke the debugger by clicking the Debug icon in the toolbar in the Project painter, or by right-clicking on the project in the System Tree and selecting the Debug menu item. If a Windows Forms application is already running, you can attach the debugger to the running process.

The debugger has almost the same operations for .NET targets as for standard PowerBuilder targets. 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. In both .NET and standard applications, you can break into the debugger when an exception is thrown (see "Breaking into the debugger when an exception is thrown" on page 13).

For more information, see "Debugging and Troubleshooting" in *Deploying Applications and Components to .NET*.

Conditional compilation

You can use conditional compilation to differentiate among target types when you are developing an application that you plan to deploy to more than one platform. PowerBuilder provides five preprocessor symbols for different target types. You can wrap code that should be parsed only in a specific target type in a #if defined *symbol*...#end if statement. There is also a DEBUG symbol that you can use to tag code that should only be compiled in a debug build.

For more information, see "About conditional compilation" in *Deploying Applications and Components to .NET* and "Conditional compilation" in the *PowerScript Reference*.

.NET language interop

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 .NET targets and save development time.

.NET language interoperability makes it possible to consume .NET classes and methods in PowerBuilder .NET targets. 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.

For more information, see "Referencing .NET classes in PowerScript" in *Deploying Applications and Components to .NET* and ".NET assembly import" next.

.NET assembly import

You can import .NET assemblies into .NET targets from the .NET Assemblies tab page in the Properties dialog box for the target.

Click the Browse button to open the Browse for a .NET Assembly dialog box, from which you can browse to import private assemblies with the *.dll*, *.tlb*, *.olb*, *.ocx*, or *.exe* extension. To import an assembly, select it and click Open. To import multiple assemblies, you must select and import them one at a time.

Click the Add button to open the Import .NET Assembly dialog box, from which you can import a shared assembly into your target. Assemblies must have a strong name. To import an assembly, select it and click OK. To import multiple assemblies, you must select and import them one at a time.

You can also use the Import .NET Assembly dialog box to import recently used assemblies.

System Tree display

The System Tree shows the classes, methods, structures, and enumerations for C# assemblies that you import into your .NET targets. However, a language-related limitation affecting managed C++ assemblies prevents the System Tree from displaying members of classes, structures, and enumeration types. It also causes managed C++ classes to display as structures.

By default, the full name of each class in an assembly is displayed in the System Tree. If you prefer to show only the final name, add the following line to the `[PB]` section of your *pb.ini* file:

```
SystemTree_DotNetFullName=0
```

For example, with this setting the Microsoft.SqlServer.Server.DataAccessKind class in *System.Data.dll* displays as DataAccessKind. You can right-click the class and select Properties from the pop-up menu to display the full class name.

PowerBuilder user interface and usability features

The following changes to the PowerBuilder user interface have been made in PowerBuilder 11:

- New target types
- Setting the current target
- Project painter enhancements
- System Tree enhancements
- Output window enhancements
- Resizable dialog boxes
- Updated menus and toolbars
- PowerBuilder Application Server Plug-in
- Application Server Profiles dialog box available in wizards
- Suppressing warning messages for objects in source control
- Setting AutoScript options
- Modified date displays for copy and paste
- Target field in Browser resized
- Edit styles are created in the Object Details view
- Breaking into the debugger when an exception is thrown
- Selecting EAServer components for debugging

New target types

In previous versions of PowerBuilder, you could create only two kinds of target: PowerScript targets and Web (JSP) targets. When you selected EAServer Component, COM/COM+ Component, or Automation Server on the Target page in the New dialog box, you created a component in a PowerScript application target. To enhance usability in the PowerBuilder 11 development environment, the items on the Target page now create a specific type of target.

PowerBuilder 11 has several new target types. The Application Server Component wizard creates a component that you can deploy to a J2EE server. For more information, see “PowerBuilder Application Server Plug-in” on page 11. There are four .NET wizards. For more information, see “.NET deployment features” on page 2.

The existing EAServer Component item in the New>Target page displays a wizard that creates an EAServer target, not an Application target as in previous versions of PowerBuilder.

The Automation Server and COM/COM+ wizards have been removed from the Target page in the New dialog box. To create COM/COM+ or Automation Server components, you must first create an application target, and then use the wizard on the Object page to create the component.

Each of the new targets has an associated project type. You provide deploy, run, and debug instructions in the project object. If a library in your target contains projects for a different target type, they do not display in the System Tree.

Setting the current target

Your current target displays in the System Tree using a bold font. The current target is the default target used in the New dialog box and for Run and Debug. The current target is set whenever you:

- Invoke an action in the System Tree, Library painter, or main menu that affects a target or a child of a target , such as Build, Migrate, Run, or Debug. Some actions, such as Search and Migrate, display a dialog box. If you cancel the action by clicking the Cancel button in the dialog box, the current target is not changed.
- Open an object painter or a file in a JSP target.
- Change the active object painter.

If you prefer to set the current target explicitly using the Set as Current Target pop-up menu item for the target in the System Tree or the File>Set Current Target menu item, clear the Automatically Set Current Target check box on the Workspaces tab page in the System Options dialog box.

Project painter enhancements

In Project painters, the Select Objects and Properties dialog boxes have been removed. Property tab pages display in the painter workspace. The choices that you used to make in the Select Objects dialog box can now be made on the Components tab page in the Project painter workspace. To get online Help, right-click in the painter workspace to display the Help pop-up menu.

Most Project painters have a Run tab page. The settings you can make on this page depend on the target. In most projects you can specify command-line arguments and change the directory in which the application starts. In a Web Forms target, you can choose the browser that the application will run in. For EAServer and application server components, you can choose to start a client application that calls the component or run the client application in the debugger in another instance of PowerBuilder.

System Tree enhancements

There are several enhancements to the System Tree in PowerBuilder 11:

- You can double-click an event or function in the System Tree to open its script in a painter.
- Events and functions that have scripts are identified by a script icon.
- If you import a .NET assembly into a .NET project, the assembly displays in the System Tree and can be expanded to display its classes and methods.

Output window enhancements

The Output window displays the messages that displayed in one pane in previous versions of PowerBuilder in separate tabs for different message types.

When you start a new PowerBuilder session, the Output window has a single tab, Default. New tabs are added as you perform operations. Tabs display in the order in which they are created and remain in the Output window for the rest of the PowerBuilder session. To clear the output from the tabs when you start a new build, make sure that the Automatically Clear Output Window check box on the General page of the System Options dialog box is selected.

Tab	Contents
Default	General information about the progress of full or incremental builds and project deployment
Debug	Debugger output, such as the paths of assemblies loaded to support .NET debugging
Errors	Messages that indicate problems that prevent the build or deploy process from completing successfully
Warnings	Warning and informational messages
Search	Output from search operations
Unsupported features	For .NET targets, names and locations of features not supported in the target type

When a message refers to a specific object or script, you can double-click the message to open the object.

Resizable dialog boxes

Dialog boxes in the PowerBuilder 11 development environment are resizable. For example, select File>New to display the New dialog box and then click the Project tab. In PowerBuilder 11, you can resize the New dialog box to display all the project types with no scrolling. In PowerBuilder 10.5, you had to scroll down to display all the project types.

Updated menus and toolbars

Menus and toolbars in the PowerBuilder 11 development environment use the contemporary style introduced for deployed applications in PowerBuilder 10.5.

PowerBuilder Application Server Plug-in

The PowerBuilder Application Server Plug-In is a standalone product that enables PowerBuilder users to deploy components to and write clients for third-party application servers. Supported servers in the first release of the product are WebSphere 6.1, WebLogic 9.2, and JBoss 4.0.4.

The EAServer Profiles dialog box has been renamed “Application Server Profiles” and can now be used to specify profiles for other J2EE servers.

New icons for the Application Server Component generator have been added to the Target, PB Object, and Project pages in the New dialog box in PowerBuilder Enterprise. New icons for the Application Server Proxy generator have been added to the Project page.The new wizards and their usage are very similar to the related EAServer wizards.

Some new features added to EAServer components also apply to application server components. For example, they can be deployed as EJB 2.1 Web services, and you can specify security roles and custom EJB properties in the Project painter.

These changes are also in PowerBuilder 10.5.1.

Application Server Profiles dialog box available in wizards

In the EAServer and Application Server Component and Proxy wizards, there is now a Manage Profiles button on the Choose EAServer Profile or Choose Application Server Profile page. Clicking this button opens the Application Server Profiles dialog box so that you can add a new profile or modify an existing profile without leaving the wizard.

The Profile Name in the Edit Application Server Profile dialog box cannot be edited. This is because the name is now stored in the project object along with the other properties of the profile. If the profile name cannot be found in the registry when the project is deployed, the description in the project object is used.

Suppressing warning messages for objects in source control

Although you can open objects in a PowerBuilder painter when they are checked in to source control, until you check them out again, any changes you make to those objects cannot be saved. By default, when you try to open an object under source control, PowerBuilder provides a warning message to let you know when the object is not checked out. You can avoid this type of warning message by clearing the "Suppress prompts to overwrite read-only files" check box on the Source Control tab of the Workspace Properties dialog box.

If you did not change the default, you can still select a check box on the first warning message that displays. After you select the "Do not display this message again" check box in a warning message box and click Yes, the check box on the Source Control tab is automatically cleared. This prevents warning messages from displaying the next time you open objects that are checked in to source control. Although warning messages do not display, you still cannot save any changes you make to these objects in a PowerBuilder painter.

Setting AutoScript options

In painters with a Script view, you can set AutoScript options from the Design>Options menu. A new Show Return Types check box has been added to the Options page.

When you paste statements into a script using the Edit>Paste Special menu item, prototype values display in the syntax to indicate conditions or actions. By default, the statements are pasted in lowercase. To paste statements in uppercase, add the following line to the [PB] section of the *PB.INI* file:

```
PasteLowercase=0
```

This *PB.INI* setting now applies to AutoScript as well as Paste Special.

Modified date displays for copy and paste

The message prompt that displays when you copy an object from one PBL to another PBL that already has an object of that name now includes the modification date and time to help you determine whether to confirm the copy.

Target field in Browser resized

The Target field at the top of the Browser has been resized to the full width of the Browser to make it easier to see which target is selected.

Edit styles are created in the Object Details view

When you create or modify an edit style in the Database painter, you specify the style in the Object Details view, as you do for Display Formats and Validation Rules, instead of in a separate dialog box. You can now also create a custom edit style for the InkEdit edit style.

Breaking into the debugger when an exception is thrown

When an application throws an exception while it is being debugged, the debugger sees the exception before the program has a chance to handle it.The debugger can allow the program to continue or it can handle the exception. This is usually referred to as the debugger's first chance to handle the exception. If the debugger does not handle the exception, the program sees the exception. If the program does not handle the exception, the debugger gets a second chance to handle it.

You can control whether the debugger handles first chance exceptions in the Exception Setting dialog box. To open the dialog box, open the Debugger and select Exceptions from the Debug menu. By default, all exceptions inherit from their parent and all are set to Continue.

When one of these exceptions is thrown, a dialog box displays so that you can choose whether to open the debugger or pass the exception to the program.

Selecting EAServer components for debugging

When you debug an EAServer target, the set of components that can be debugged is determined from the project. The set includes all components selected on the Components page in the Project painter for which the Remote Debugging check box is selected. If you want to select a different set of components or debug components from more than one package, select Debug>Select Components from the menu bar in the EAServer debugger or click the Select Components button on the PainterBar.

New features for Window controls

New properties for the Window control allow you to add some special effects to your applications:

- Opening and closing windows with an animated effect
- Transparency for windows

You set the properties on the General page in the window's Properties view.

Opening and closing windows with an animated effect

You can use a special effect when a window opens or closes. Effects include fading in or out, opening from the center, and sliding or rolling from the top, bottom, left, or right. Set the AnimationTime property to between 1 and 5000 milliseconds to specify how long the animation effect takes to complete.

For example, if your application displays a splash screen while the application's main window is initializing, you can set the CloseAnimation property to have the window fade out rather than just disappearing when the application is initialized or after a timeout:

```
w_splash.CloseAnimation = FadeAnimation!
```

For more information, see the OpenAnimation , CloseAnimation, and AnimationTime properties for the Window control in *Objects and Controls* or the online Help.

Transparency for windows

You can specify a value between 1 and 100% for the Transparency property of a window. This property is useful if you want a non-modal dialog box to remain visible but become semi-transparent when it loses focus.

For more information, see the Transparency property for the Window control in *Objects and Controls* or the online Help.

DataWindow features

The following new DataWindow features are available in PowerBuilder 11:

- TreeView DataWindow features
- Web service as a DataWindow data source
- Remember DataWindow retrieval arguments
- UseEllipsis DataWindow object property

TreeView DataWindow features

The following changes have been made to the TreeView DataWindow:

- You can hide tree nodes in the detail band by setting a height of 0 for the detail.
- You can move rows in a TreeView and they will retain their expanded or collapsed state.
- You no longer need to define user-defined events for the Collapsing, Collapsed, Expanding, and Expanded events—they are now standard events on the DataWindow object.

The TreeView DataWindow style can also be used to create Web DataWindows in .NET Web Forms applications in PowerBuilder 11.0. (The TreeView Web DataWindow style for Web DataWindows is not supported in JSP targets.)

The TreeView Web DataWindow supports most of the features available in the TreeView DataWindow in Windows applications. The ShowConnectLines and ShowLeafNodeConnectLines properties are not supported in the current release for performance reasons.

Image files used for tree node icons, like other image files, must be deployed to the Web site with the PBL. To do this, select the icon on the Resource Files page in the Project painter.

Web service as a DataWindow data source

In PowerBuilder 11 you can use a Web service as the data source for DataWindow objects with any presentation style except RichText and OLE.

For more information on using a Web service data source for DataWindow objects, see the "Defining DataWindow Objects" chapter in the PowerBuilder *User's Guide*.

For information on updating DataWindow objects using a Web service data source, see the "Controlling Updates in DataWindow Objects" chapter in the PowerBuilder *User's Guide*.

Remember DataWindow retrieval arguments

In previous versions of PowerBuilder, when you specified retrieval arguments for a DataWindow object in the Specify Retrieval Arguments dialog box, the retrieval argument values that you specified were not remembered by the DataWindow object.

To enhance usability, the Specify Retrieval Arguments dialog box now has a Remember retrieval arguments check box. When the Remember retrieval arguments check box is selected, the data values are saved and the check box remains selected when you press the OK button. Clearing the Remember retrieval arguments check box and clicking OK removes the data values. This feature is only available in the development environment.

Not for nested and Composite DataWindows
You cannot use the Remember Retrieval Arguments feature in a nested or Composite DataWindow, so the Remember Retrieval Arguments check box is grayed out.

In a new DataWindow, the Remember retrieval arguments check box is disabled until you save the DataWindow and name it. You must close and reopen the DataWindow object or the Preview view to enable the checkbox.

UseEllipsis DataWindow object property

If a column with the Edit or EditMask edit style contains character data that is too long for the display column in the DataWindow, the data is truncated. You can choose to display an ellipsis at the end of the truncated data. To do so, select the Use Ellipsis check box on the Format page in the Properties view or specify the UseEllipsis DataWindow object property in a script:

```
dw1.Object.col1.Edit.UseEllipsis = Yes
dw1.Modify("col1.Edit.UseEllipsis=Yes")

dw1.Object.col1.EditMask.UseEllipsis = Yes
dw1.Modify("col1.EditMask.UseEllipsis=Yes")
```

For displayed text, if the end of the string does not fit in the rectangle, it is truncated and the ellipsis is displayed. The ellipsis does not display when the column has focus.

The property is ignored if you:

- Check Autosize Height on the Position page or set the Height.Autosize property in a script.
- Specify an expression for the Escapement property on the Font page to rotate the text or set the Font.Escapement property in a script.

The UseEllipsis DataWindow object property is not supported in Web Forms applications.

Database connectivity features

The following database connectivity features are available in PowerBuilder 11:

- Enhanced support for MobiLink synchronization
- Support for Adaptive Server Enterprise 15
- Support for Adaptive Server 15 unsigned datatypes
- Support for Microsoft SQL Server 2005
- Support for HA event notification in Oracle 10g

Enhanced support for MobiLink synchronization

MobiLink is a session-based synchronization system that allows two-way synchronization between a main database, called the consolidated database, and multiple remote databases. The ASA MobiLink Synchronization wizard on the Database tab of the New dialog box creates objects that facilitate control of database synchronization from a PowerBuilder application.

Modifications to the wizard

In PowerBuilder 11, system objects have been added to enable MobiLink functionality to work with .NET applications. Although the ASA MobiLink Synchronization wizard still generates objects that facilitate control of database synchronization from PowerBuilder applications, the main nonvisual object generated by the wizard is now an instance of the MLSync system object (that inherits from the MLSynchronization base class).

The wizard no longer generates a global structure object for storage of synchronization parameters entered by the user, but the synchronization window that it generates uses the SyncParm system structure instead. These differences apply even in targets that are not deployed to .NET platforms.

Migrating MobiLink objects

In PowerBuilder 10.5 MobiLink applications that you migrate to PowerBuilder 11, Sybase strongly recommends that you rerun the MobiLink wizard and generate new synchronization objects. You must rerun the wizard if you are deploying to .NET Windows Forms targets.

Synchronization applications that you migrate can still work with standard PowerBuilder applications without rerunning the wizard, but you must modify the library object references in the pb_run_dbmlsync and pb_cancel_dbmlsync functions of the wizard-generated nvo_*appname*_mlsync object. For PowerBuilder 11, the referenced library for these functions must be *pbodb110.dll*, not *pbvm105.dll*.

The following table shows objects that can be generated by the wizard, listed by their default object names, where *appname* stands for the name of the current application.

Default name	Description
nvo_*appname*_mlsync	An instance of the MLSync standard class user object that starts synchronization from the remote client. In PowerBuilder 10.5, this was a simple nonvisual user object. **Name change for PowerBuilder 11** The default suffix "_mlsync" replaces the "_sync" default suffix used by earlier versions of the wizard.
gf_*appname*_sync	Global function that instantiates nvo_*appname*_mlsync to start the synchronization. This function includes the logic to start the synchronization with or without a feedback window.
w_*appname*_syncprogress	Optional feedback window that can be used to display synchronization status to the client. **Name change for PowerBuilder 11** The default suffix "syncprogress" replaces the "_sync" default suffix used by earlier versions of the the wizard.
gf_*appname*_configure_sync	Optional global function that calls the w_*appname*_sync_options window, which allows the user to configure the dbmlsync client.
w_*appname*_sync_options	Optional window that allows the application user to change connection arguments at runtime.

Creating an instance of MLSync

With the new MLSync system object in PowerBuilder 11, you no longer have to use the MobiLink Synchronization Wizard to create a nonvisual object that launches Dbmlsync.exe. You can include an MLSync object in your applications:

- Programmatically with PowerScript

  For an example showing how to add an MLSync object programmatically, see the chapter on MobiLink Synchronization in *Application Techniques*.

- By selecting it from the New dialog box

For more information about MobiLink synchronization, see the chapter on "Managing the Database" in the *User's Guide* and the chapter on "Using MobiLink Synchronization" in *Application Techniques*. For more information on system objects related to synchronization, and their functions, events, and properties, see MLSynchronization, MLSync, and SyncParm in the online Help.

Auxiliary objects for MobiLink synchronization

If you create an instance of MLSync by PowerScript code or from the New dialog box, you should also consider using auxiliary objects that can be generated automatically by the wizard, or that you can customize in the PowerBuilder Window painter.

The chapter on MobiLink synchronization in *Application Techniques* includes a section describing the default progress and options windows and suggestions for customizing them.

Support for Adaptive Server Enterprise 15

A new database interface, the ASE interface, has been added to support Adaptive Server® Enterprise 15 and later releases. This interface is identical to the existing SYC interface, except that:

- It currently supports only Adaptive Server 15 (the Release database parameter can only be set to 15).
- It supports large identifiers with up to 128 characters. This support is not available in the SYC interface.

To use this interface, the Adaptive Server 15 client must be installed on the client computer. The ASE interface will also be available in PowerBuilder 10.5.1.

Support for Adaptive Server 15 unsigned datatypes

In PowerBuilder 10.5, support for Adaptive Server 15 unsigned datatypes was added for the SYC and JDBC interfaces. In PowerBuilder 11, this support is also available for the ODBC interface and the new ASE interface.

Support for Microsoft SQL Server 2005

A new database interface, the SNC interface, has been added to support Microsoft SQL Server. The new interface uses the SQL Server 2005 native client (*sqlncli.h* and *sqlncli.dll*) on the client side and connects using OLE DB.

PBODB initialization file not used
Connections made directly through OLE DB use the PBODB initialization file to set some parameters, but connections made using the SNC interface do not depend on the PBODB initialization file.

The SNC interface can be used to connect to SQL Server 2005 and SQL Server 2000.

For SQL Server 2000, the SQL client SDK was provided with the Microsoft Database Access Components (MDAC). MDAC does not support new features in SQL Server 2005. To use the SNC interface, the SQL Server 2005 SQL Native Client and the .NET Framework 2.0 must be installed on the client computer.

For more information, see the chapter on Microsoft SQL Server in *Connecting to Your Database*.

Support for HA event notification in Oracle 10g

Oracle Real Application Clusters (RAC) is a cluster database that uses a shared cache architecture. In Oracle 10*g* Release 2, a High Availability (HA) client connected to an RAC database can register a callback to indicate that it wants the server to notify it in case of a database failure event that affects a connection made by the client.

To take advantage of this feature, PowerBuilder users can script a new event, DBNotification, that has been added to the Transaction object. For more information, see the description of the DBNotification event and the HANotification database parameter in the online Help.

New and modified functions

Byte array conversion functions

A new PowerScript function, GetByteArray, and a new syntax for the Blob function have been added to enable conversion between blobs and byte arrays.

The syntaxes are:

```
Blob ( any stringorbytearray ) returns blob
GetByteArray ( blob input ) returns Any
```

Two ways to use trim functions

A new overloaded version of the Trim, LeftTrim, and RightTrim functions has been added. By default, these functions trim only space characters. To trim all types of white space characters, such as tabs and carriage returns, set the optional second argument to true. The default is false.

Using \s in a filter expression

You can now append `\s` to the filter expression you use with the SetFilter DataWindow function if you want to compare strings in ASCII order instead of dictionary order. For example, the following expression shows only rows in which column 2 begins with a or b, because the ASCII values of uppercase letters are lower than the ASCII values of lowercase letters:

```
#2 >= 'a' and #2 < 'c' \s
```

Without the `\s`, rows in which column 2 begins with A, a, B, or b would display.

See the function descriptions in the online Help for more information.