New Features PowerBuilder® 11.5

Document ID: DC00357-01-1150-01

Last revised: September 2008

Copyright 2008 by Sybase, Inc. All rights reserved. Sybase trademarks can be viewed at the Sybase trademarks page at http://www.sybase.com/detail?id=1011207. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

DataWindow display enhancements

In PowerBuilder 11.5, you can enhance the DataWindow® with gradients, transparency settings, and pictures. New object properties, including Brushmode and Picture, allow you to change the appearance of the DataWindow background. Using these new properties gives your applications a modern look and feel. For example, you can use the Brushmode property to create background gradient effects that give the DataWindow more visual interest.

Gradients display transitions from one color to another. Using Brushmode, you can choose from several different styles of gradient effects (horizontal, vertical, angled, radial) and then set other properties to customize the appearance of the gradient and transitions. If you choose to use a picture, then you would use the Picture properties to determine the appearance of the picture, whether it is original size, stretched to fit, or tiled in any way. You can adjust the transparency settings for both colors and pictures.

For examples of how to use these new properties, refer to “Enhancing DataWindow objects” in the *Users Guide*. For more information on the new DataWindow display properties, see the online Help for the following items:

- Brushmode.property
- Picture.property
- Gradient.property
- Transparency (DataWindow objects)

RichText Edit style for DataWindow columns

In PowerBuilder 11.5, you can use the RichText edit style to display column data in a rich text format, and to use different fonts and colors in the same data field.

Columns that you format with the RichText edit style require considerably more storage space than columns with plain text edit styles. Therefore you should set a minimum of 1 KB for the column width. Otherwise, you can use the RichText edit style with columns that have large text datatypes.

Existing methods and events
You can use the GetText method to return text with its rich text formatting and the SetText method to set text with rich text formatting. However, the data arguments for the ItemChanged and ItemError events return text from rich text edit columns without the rich text formatting.

By default, whenever a column with the RichText edit style is edited in the Preview view or at runtime, a font toolbar displays. The font toolbar disappears when the column loses focus. The toolbar can be moved and remembers its last position. You can modify the RichTextToolbarActivation constant on a DataWindow control to display the font toolbar whenever a DataWindow object containing columns with the RichText edit style has focus—whether or not this type of column is selected. You can also modify the constant so that the font toolbar never appears.

DataWindow property	Datatype	Description
RichTextToolbar Activation	RichTextTool barActivation (enumerated)	Specifies when the default font toolbar appears for a DataWindow object that has columns with the RichText edit style. Values are: RichTextToolbarActivationAlways! RichTextToolbarActivationNever! RichTextToolbarActivationOnEdit! (default)

You can customize the toolbar by taking advantage of the properties, events, and methods introduced to support the RichText edit style. If you want to use a customized font toolbar instead of the default toolbar, you must set the RichTextToolbarActivation constant to RichTextToolbarActivationNever!.

A new DataWindow expression allows you to strip the RTF formatting of a RichText column. For details, see StripRTF in the online Help.

For more information on RichText column properties and methods, see the online Help for the following items:

RichTextToolbarActivation	RichTextCurrentStyleChanged
RichTextLoseFocus	RichTextLimitError
GetRichTextAlign	GetRichTextColor
GetRichTextFaceName	GetRichTextSize
GetRichTextStyle	SetRichTextAlign
SetRichTextColor	SetRichTextFaceName
SetRichTextSize	SetRichTextStyle

3D Graph styles

PowerBuilder 11.5 adds 3D rendering to display the 3D graphs (Pie3D, Bar3D, Column3D, Line3D, and Area3D) with a more sophisticated look. The new graph styles let you use data item translucency instead of the overlays used by original 3D graph styles. However, you can still use the 3D graphs with the original rendering style if you want.

The new graph rendering style is supported for standalone graph controls and for graph controls in a DataWindow object. PowerBuilder uses the following functions to support the new graph styles in standalone graph controls:

GetDataLabelling	SetDataLabelling
GetDataTransparency	SetDataTransparency
GetSeriesLabelling	SetSeriesLabelling
GetSeriesTransparency	SetSeriesTransparency

PowerBuilder uses the same functions to support the new graph styles in DataWindow controls, but with an extra *graphcontol* argument.

For more information, look up the function in the online Help.

DirectX runtime

The new 3D rendering depends on the DirectX runtime, which will be installed the first time the user selects the feature. After the Render3D check box is initially selected, the DirectX installer will launch. If the check box is selected for an unsupported style, nothing happens until one of the supported 3D graphs is selected. If you opt out of the installation, the Render3D property is ignored.

If you install DirectX on the runtime computer but selecting the Render3D check box does not change the appearance of the graph, it is possible that the graphics card does not support DirectX.

You can check whether DirectX is supported by running dxdiag.exe. This file is typically installed in the *Windows\System32* directory. The Display tab of the DirectX Diagnostic Tool indicates whether Direct3D is enabled.

Unsupported properties

The following properties are not supported in the new 3D graph styles.

- Axis: ShadeBackEdge, MajorTic, MajorGridLine, DropLines, MinorDivisions, MinorTic, MinorGridLine, PrimaryLine, SecondaryLine, OriginLine, Frame
- Text: Alignment for axis labels and text.

Tooltips for DataWindow columns and controls

Tooltips display text when the pointer pauses over a DataWindow column or control. This text can be used to explain the purpose of the column or control. To use this feature, select the column or control for which you want to create a tooltip and then select the Tooltip tab in the Properties view. You can use the tab to specify:

- Text for the tooltip
- Title for the tooltip
- Color of the background and text
- Icon for the tooltip
- Delay before the tooltip appears and disappears
- Whether the tooltip appears as a rectangle or callout bubble

For more information, see Tooltip.property in the online Help.

Design-time support for picture transparency

PowerBuilder 11.5 introduces design-time support for the existing transparentcolor DataWindow property. To use this feature, right-click on the color of the picture that you want to make transparent and then select Make Color Transparent from the context pop-up menu. The Transparent Color field populates with the number for the color you selected. Make sure that Transparency is not set to zero.

You can also enter the number for the color by calculating the value with the formula found in the topic "Using colors with display formats" in the *Objects and Controls* book.

You can save the picture with its transparency settings to a WMF, EMF, or PSR file using the Save Rows As menu item.

This feature does not apply to:

- Colors that are already transparent
- Windows Meta Files (WMF)
- RichText or OLE processing styles
- Web Forms applications

- Printed reports or DataWindow objects
- Save Rows As PDF and other formats except PSR, EMF, and WMF

New DataWindow object properties

PowerBuilder 11.5 allows you to manipulate the presentation of DataWindow objects and their columns and controls in a number of new ways—through the definition of gradient backgrounds, picture backgrounds, scalable transparency settings, and tooltip properties for columns and controls.

For information on these new properties, see the *DataWindow Reference* in the online Help.

New DataWindow object background properties

- Bandname.property (gradient properties)
- Brushmode
- Gradient.property
- Picture.property
- Transparency (DataWindow objects)

New column and control properties

- Background.property
- Render3D
- Tooltip.property
- Transparency (columns and controls)
- Transparency (picture controls in DataWindows)

New DataWindow events and methods

In addition to the new properties listed in "New DataWindow object properties" on page 6, the enhanced DataWindow functionality in PowerBuilder 11.5 uses the following events and methods.

New events

RichTextCurrentStyleChanged
RichTextLimitError
RichTextLoseFocus

New methods

GetDataLabelling (Graphs in DataWindows)
GetDataTransparency (Graphs in DataWindows)
GetSeriesLabelling (Graphs in DataWindows)
GetSeriesTransparency (Graphs in DataWindows)
SetDataLabelling (Graphs in DataWindows)
SetDataTransparency (Graphs in DataWindows)
SetSeriesLabelling (Graphs in DataWindows)
SetSeriesTransparency (Graphs in DataWindows)
GetRichTextAlign
GetRichTextColor
GetRichTextFaceName
GetRichTextSize
GetRichTextStyle
SetRichTextAlign
SetRichTextColor
SetRichTextFaceName
SetRichTextSize
SetRichTextStyle

The RichText functionality for DataWindow columns also uses the new constant, RichTextToolbarActivation.

PNG support

PowerBuilder 11.5 supports the Portable Networks Graphics (PNG) file format for the images that you use in menus, toolbars, treeview controls, and DataWindow objects. PNG images display at design time as they do at runtime. Transparent portions of a PNG image display the background color or color gradient of the control to which they are assigned when the control has a background color or color gradient property.

In a Picture control For picture controls without a background color property, transparent portions of a PNG image display the background color of the parent object or control. For example, the PowerBuilder Picture control does not have a background color property. If you assign a transparent PNG image to the Picture control, the transparent portions display the background color of the window or object that contains the Picture control.

In a PictureButton control Unlike the Picture control, the PictureButton control does have a background color property in PowerBuilder. If you assign a transparent PNG image to the PictureButton control, it displays with the background color assigned to the PictureButton control. If you want the PNG image in a PictureButton control to blend in with the background of the container object for the PictureButton control, you must set the PictureButton's BackColor property to the background color of its container object.

Covering other controls or images If you display a transparent PNG image in a Picture control on a window, and the Picture control covers part of a ListBox control (or other noncontainer control or object), the transparent portion of the PNG image clips the control or object it covers and displays with the background color of its container object. If the container object, such as a window, includes a background image in addition to a background color, the PNG image displays the background color only and clips any background image that it covers.

Supported target types and exceptions You can deploy PNG images with standard PowerBuilder targets and with .NET Windows Forms or Web Forms targets. However, PNG images cannot be selected in DataWindow objects with the RichText or OLE presentation style. Also, if you export a DataWindow object as a PDF file using the XSLFOP method, PNG images in the DataWindow are not saved in the generated PDF. PNG images with transparency settings are fully supported in Internet Explorer 7.0 browsers, but earlier versions of the browser cannot display the transparency effects of these images.

PNG images are drawn with GDI+ functions rather than with bitmap functions. Therefore the Map3DColors property for Picture and PictureButton controls is ignored when you select a PNG file for the picture image. PNG images are also kept locked while the image is active. This is in contrast to bitmap images, which are released once the image is loaded.

.NET target enhancements

PowerBuilder 11.5 includes the following enhancements for .NET targets:

- Security enhancements
- Strong-named assemblies
- Added support for remote deployment (Vista and IIS 7)
- Support for shared objects in .NET targets

Security enhancements

Code access security

PowerBuilder 11.5 applications and components can run in partial trust environments when they are constrained by .NET code access security (CAS) configurations. PowerBuilder lets you configure CAS security zones (sandboxes) for .NET Web Forms, .NET Web Service, .NET Windows Forms and SmartClient projects to minimize the amount of trust required before application or component code is run by an end user.

For .NET Web Forms and Web Service projects, you can also modify the *Web.config* file to support security zones after you deploy the project. In PowerBuilder 11.5, the .NET assemblies that you create by building and deploying .NET Assembly projects are run with the security permissions of the calling application or component. In earlier releases, PowerBuilder applications and components could run with full trust only.

Security tab enhancements

A radio button group field on the Project painter's Security tab allows you to select full trust or a customized trust option. For Windows Forms applications, you can also select local intranet trust or internet trust. A list box below the radio button group allows you to select or display the permissions you want to include or exclude when you select local intranet trust, internet trust, or the custom option. (If you select full trust, the list box is disabled.)

Viewing default permissions in the .NET Framework Configuration tool
The list of permissions that display in the Security tab list box is the same as the list in the .NET Framework 2.0 SDK Configuration tool that you open from the Administrative Tools folder in the Control Panel for your computer. To see the permission settings in the configuration tool, select My Computer>Runtime Security Policy>Machine>Permission Sets>Everything in the treeview in the left pane of the configuration tool, then if the list is not already displayed, click View Permissions in the right pane.

If you modify a permission after selecting the local intranet or internet options, PowerBuilder automatically treats the selected permissions as custom selections, but does not modify the selected radio button option. This allows you to click the Reset button above the list box to change back to the default local intranet or internet permission settings. Clicking the Detail button (to the left of the Reset button) opens the Custom Permissions dialog box that allows you to enter custom permissions in XML format. The Reset and Detail buttons are disabled only when you select the Full Trust radio button option.

For Smart Client applications, the permission information is stored in the manifest file that you deploy with your application. When users run a Smart Client application, the application loader process loads the manifest file and creates a sandbox where the application is hosted.

For standard Windows Forms applications, the sandbox allows you to run the application with the permissions you define on the Project painter Security tab when the applications are run from the PowerBuilder IDE. When a user starts Windows Forms applications from a file explorer or by entering a UNC address (such as \\server\myapp\myapp.exe), the security policies set by the current user's system are applied and the Security tab permission settings are ignored.

Custom security settings

For information on custom security permissions, see the appendix to the *Deploying Applications and Components to .NET* book and the Microsoft Web site at http://msdn.microsoft.com/en-us/library/system.security.permissions.aspx.

Permission error messages

If your .NET application attempts to perform an operation that is not allowed by the security policy, the Microsoft .NET Framework throws a runtime exception. For example, the default local intranet trust level has no file input or output (File IO) permissions. If your application runs with this security setting and tries to perform a File IO operation, the .NET Framework issues a File Operation exception.

You can catch .NET security exceptions using .NET interoperability code blocks in your PowerScript code:

```
#if defined PBDOTNET then
   try
      ClassDefinition cd_windef
      cd_windef = FindClassDefinition("w_1")
      messagebox("w_1's class
      definition",cd_windef.DataTypeOf)
   catch(System.Security.SecurityException ex)
       messagebox("",ex.Message)
   end try
#end if
```

All .NET targets must include a reference to the *mscorlib.dll* .NET Framework assembly in order to catch a System.Security.SecurityException exception. The *PBTrace.log* files that PowerBuilder Windows Forms and Web Forms applications generate by default contain detailed descriptions of any security exceptions that occur while the applications are running. PowerBuilder .NET Web Service components also generate *PBTrace.log* files that log critical security exceptions by default.

If you do not catch the exception when it is thrown, the PowerScript SystemError event is triggered. For Windows Forms applications, a default .NET exception message displays if you do not catch the exception or include code to handle the SystemError event. The exception message includes buttons enabling the user to show details of the error, continue running the applicatiion, or immediately quit the application. Under the same conditions for Web Forms applications, a system error dialog box displays instead of the .NET exception dialog box, and the application is terminated.

Debugging and tracing with specified security settings

You can debug and run .NET applications and components from the PowerBuilder IDE with specified security settings. To support this capability in Windows Forms applications, PowerBuilder creates a hosting process in the same directory as the application executable. The hosting process creates a domain with the CAS setting before it loads the application assemblies. (The CAS settings generated in the *Web.config* file determine the security permissions used by .NET Web Forms applications and .NET Web Service components.)

If your .NET application attempts to perform an operation not allowed by the specified security setting, an exception is issued in the IDE.

Strong-named assemblies

PowerBuilder 11.5 can generate strong-named assemblies from all .NET Project painters.

A strong name consists of an assembly's identity—its simple text name, version number, and culture information (when provided)—plus a public key and digital signature. It is generated from an assembly file using the corresponding private key. The assembly file contains the assembly manifest that includes the names and hashes of all the files that make up the assembly.

Project painter Sign tab

PowerBuilder 11.5 includes a new Sign tab in the Project painter for all .NET application and component projects. The Assembly group box on the Sign tab allows you to attach strong name key files to the assemblies that the.NET projects generate. The Assembly group box contains the following fields:

Assembly group box field	Description
Sign the assembly	Select this check box to enable the "Choose a strong name key file" single line edit box, the browse and New buttons, and the "Delay sign only" check box.
Choose a strong name key file	Name of the key file you want to attach to the generated assembly. This field is associated with a browse (ellipsis) button and a New button. The browse button opens a Select File dialog box where you can select a key file with the *.snk* or *.pfx* extension. The New button lets you create a key file with the *.snk* extension. PowerBuilder uses the *Sn.exe* tool from the .NET Framework SDK to create the key file.
Delay sign only	Select this check box if your company's security considerations require the step of signing the assembly to be separate from the development process. When this check box is selected, the project will not run and cannot be debugged. However, you can use the strong name tool *Sn.exe* (in the .NET Framework SDK) with the `-Vr` option to skip verification during development.
Mark the assembly with AllowPartiallyTrustedCaller Attribute (.NET Web Service and .NET Assembly projects only)	By default, a strong-named assembly does not allow its use by partially trusted code and can be called only by other assemblies that are granted full trust. However, you can select this check box to allow a strong-named assembly to be called by partially trusted code.

The Sign tab has additional fields for selecting certificate files that you publish with smart client applications.

The Intelligent Updater group box replaces the Certificate group box that was on the Publish tab of Windows Forms projects in earlier releases of PowerBuilder. It is grayed out for .NET Windows Forms projects when the Publish as smart client application check box on the General tab of the Project painter has not been selected, and it is not displayed on the Sign tabs of .NET Web Forms projects or .NET component projects (Web Service and .NET Assembly projects).

The Intelligent Updater group box contains the following fields:

Intelligent Updater field	Description
Sign the manifests	Select this check box to enable the Select from Store and Select from File buttons. Use the buttons to select a certificate from a certificate store or from your file system. If you select a valid certificate, its details display in the multiline edit box under the check box. If you do not specify a certificate, PowerBuilder attaches a test certificate automatically. Use test certificates for development only.
Select from Store	Click this button to view the certificates available in the local certificate store. Select a certificate from the Select a Certificate dialog box, then click View Certificate if you want to view its details, and click OK to select it.
Select from File	Click this button to view the certificates available in the local file system. Select a certificate with the *.snk* or *.pfx* extension from the Select File dialog box and click Open.

Error messages

If you select a strong name key file in either the Assembly or Intelligent Updater group boxes, and the key file is invalid, PowerBuilder displays a message box telling you that the key file is invalid. If the key file you select is password protected, PowerBuilder prompts you to enter the password for the key file. If you enter an incorrect password, a message box informs you that the password you entered is invalid.

Added support for remote deployment (Vista and IIS 7)

PowerBuilder 11.5 .NET targets do not require you to have IIS installed on the development computer, and remote deployment of .NET targets has been enhanced. When you deploy directly to a remote computer, system information about the deployment computer, including its OS and IIS versions, is passed to PowerBuilder through the Windows Management Instrumentation (WMI) interface.

If you deploy to an MSI setup file, and run the setup file on a deployment computer, PowerBuilder can use the Windows API to obtain information about the OS and IIS versions on that computer.

Deployment to and from Windows Vista or Windows 2008

When you run PowerBuilder on Windows Vista or Windows 2008 under a standard user account, and attempt to deploy Web Forms or Web Service projects, the User Account Control (UAC) dialog box displays. This dialog box allows you to elevate your privileges for the purpose of deployment.

Deploying .NET targets to a remote Windows Vista or Windows 2008 computer might require changes to the Windows firewall, UAC, or the Distributed Component Object Model (DCOM) settings.

Settings for	Required changes
Windows firewall	Enable exceptions for WMI and file and printer sharing
UAC (When you are not running PowerBuilder with the built-in Administrator account)	If the development and deployment computers are in the same domain, connect to the remote computer using a domain account that is in its local Administrators group. Then UAC access token filtering does not affect the domain accounts in the local Administrators group. You should not use a local, nondomain account on the remote computer because of UAC filtering, even if the account is in the Administrators group. If the development and deployment computers are in the same workgroup, UAC filtering affects the connection to the remote computer even if the account is in the Administrators group. The only exception is the native “Administrator” account of the remote computer, but you should not use this account because of security issues. Instead, you can turn off UAC on the remote computer. and if the account you use has no remote DCOM access rights, you must explicitly grant those rights to the account.
DCOM	Grant remote DCOM access, activation, and launch rights to a nondomain user account in the local Administrators group of the remote computer if that is the type of account you are using to connect to the remote computer.

IIS 7 deployment enhancement
For local or remote deployment from PowerBuilder 11.5 Web Forms or Web Service projects, you can deploy to IIS 7 servers that were not installed with IIS 6 compatibility.

Support for shared objects in .NET targets

PowerBuilder 11.5 supports shared objects in .NET Web Forms and Windows Forms targets. You can use the same shared object PowerScript functions in .NET targets that you use in standard PowerBuilder targets. You can

- Use the SharedObjectRegister function to register a nonvisual object, allowing it to be shared
- Use the SharedObjectDirectory function to retrieve the list of objects that have been registered for sharing
- Use the SharedObjectGet function to obtain a reference to a shared object instance
- Make calls to functions in the shared object
- Use the SharedObjectUnregister function to unregister a user object that was previously registered as a shared object
- Use variables in the shared object

  Each shared object has its own local copy of the shared variables.

Although you can use the PowerScript shared object functions in the .NET environment, you must take the following considerations into account before you deploy PowerBuilder .NET projects that use these functions:

- Shared objects cannot support the SQLCA global transaction object

  Each shared object must use its own transaction object instance. Typically you initialize and connect to the transaction object instance in the Constructor event of the shared object, and disconnect from the shared object in the Destructor event.
- You cannot use array declarations in shared objects that you deploy to the .NET environment
- Data from a posted call to a shared object might not be displayed before an explicit postback is issued

  When execution of a main thread is complete, it refreshes the UI and does not wait for the worker thread in a call that you post to a shared object to finish. If the worker thread inserts rows of data into a DataWindow, the UI is likely to have already been refreshed by the main thread, so the new rows will not be visible until the next time that the UI is refreshed.

Language interoperability enhancements

Support for function calls on .NET primitive and enumerated types

PowerBuilder 11.5 includes support for functions that you call on .NET primitive and enumerated types. The function calls must be made inside a conditional compilation block for a .NET target.

To support function calls on .NET primitive types, the PowerBuilder .NET compiler (pb2cs) merges the functionality of these primitive types with the functionality of corresponding PowerBuilder primitive types. This means that you can use .NET primitive types and their corresponding PowerBuilder primitive types in a similar fashion. The following example makes the same ToString function call on both the .NET System.Int32 datatype and the PowerScript long datatype:

```
System.Int32 i1
long i2
i1.ToString()
i2.ToString()
```

Exception for platform-specific primitive types
The System.IntPtr and SystemUIntPtr primitive types do not have precise corresponding types in PowerBuilder—they are always treated as long datatypes. Calling functions or modifying properties on these .NET primitive types leads to a compilation error in PowerBuilder.

For a table of equivalencies between .NET and PowerScript primitive datatypes, see the chapter on “Referencing .NET Classes in PowerScript” in the *Deploying Applications and Components to .NET* book.

Function calls are also supported on .NET enumerated types that you import to a PowerBuilder .NET target. For example, suppose we define a .NET enumerated type in a .NET assembly as follows:

```
Public enum TimeOfDay
{
    Morning = 0,
    AfterNoon,
    Evening
}
```

PowerBuilder allows you to call the ToString method on the .NET TimeOfDay enumerated type after you import it to your target:

```
#if defined PBDOTNET then
   ns1.ns2.TimeOfDay daytime
   daytime = ns1.ns2.TimeOfDay.Morning!
   daytime.ToString()
#end if
```

Using instance references to access static .NET class members

PowerBuilder 11.5 lets you use instance references to access static members of .NET classes. The following is an example of an instance reference you can use for a static member of the System.Web.HttpContext class:

```
string s
#if defined PBDOTNET then
    //OLD WAY
    //s=System.Web.HttpContext.Current.ToString()
    //NEW WAY
    System.Web.HttpContext context
    context  = create System.Web.HttpContext
    s = context.Current.ToString()
#end if
```

Database interface enhancements

PowerBuilder 11.5 includes the following database interface enhancements:

- Native driver support for Oracle 11g
- Native driver support for MS SQL Server 2008

Native driver support for Oracle 11g

The PowerBuilder 11.5 setup program optionally installs the "ORA" database driver for Oracle 11*g* connections. This driver also supports session and connection pooling.

For more detailed information, see:

- Support for session and connection pooling
- ORA driver support for other recent Oracle features
- Supported Oracle features not related to the ORA driver
- Database profile dialog box changes for the ORA driver

Support for session and connection pooling

Oracle client interface (OCI) pooling for PowerBuilder applications is created when you connect to an Oracle server for the first time. The pooling is identified by the server name and character set which are passed in the DBParm parameters SQLCA.ServerName and NLS_Charset, respectively. If two Oracle connections are connected to the same Oracle server but use different character sets, the connections must reside in different connection or session pools. All pooling-related DBParm parameters must be set before the initial database connection.

Session pooling

Session pooling means that the application creates and maintains a group of stateless sessions to the database. These sessions are passed to clients as requested. If no session is available, a new one is created. When the client is done with the session, the client releases it to the pool. With session pooling, the number of sessions in the pool can increase dynamically.

Session pooling does not support external authentication using an OS account. If a Login ID is not specified in a database connection using an existing session pool, the Login ID of the session pooling creator is used for the connection.

Connection pooling

The O90 and O10 database drivers that you can use in PowerBuilder to connect to the 9.x and 10.x versions of the Oracle DBMS support connection pooling with the DBParm parameter CNNPool. For backward compatibility purposes, this parameter is also supported by the ORA driver that you use with Oracle 11*g*. However, if the Pooling parameter is used with this driver, the CNNPool parameter is ignored.

Deciding what type of pooling to use

The following table describes the circumstances under which you should make your pooling selection:

Choose	When database sessions are
Session pooling	Stateless (reusable by middle tier threads) and the number of back-end server processes can cause database scaling problems.

Choose	When database sessions are
Connection pooling	Stateful (not reusable by middle tier threads) and the number of back-end server processes can cause database scaling problems. The number of physical connections and back-end server processes is reduced by using connection pooling. Therefore many more database sessions can be utilized for the same back-end server configuration.
No pooling	Stateful (not reusable by middle tier threads) and the number of back-end server processes will never be large enough to cause scaling issues for the database. EAServer components and MTS components do not support either type of pooling for Oracle databases.

Load balancing

The Oracle Real Application Clusters (RAC) database option allows a single database to be hosted in multiple instances on multiple nodes of the database server. This adds high availability and failover capacity to the database.

Connect time load balancing Balancing of work requests occur at two different times: connect time and runtime. Connect time load balancing occurs when a session is first created by the application. This ensures that sessions that are part of the pool are well distributed across RAC instances, and that sessions on each of the instances have a chance to execute work.

For session pools that support services at one instance only, the first available session in the pool is adequate. When the pool supports services that span multiple instances, work requests need to be distributed across instances so that more requests go to the instances with greater capacity or that provide better service.

Runtime connection load balancing You can also use runtime connection load balancing to direct work requests to the sessions in a session pool that best serve the work. Runtime connection load balancing is enabled by default when an Oracle 11.1 or higher client is connected to a release 10.2 or higher Oracle server using OCI session pooling.

The DBParm parameter, RTConnBalancing, supports the runtime connection load balancing feature. It is available only when the Pooling parameter is set to Session Pooling, and it can be set before connection only. By default, when you select Session Pooling for the pooling type, the RTConnBalancing value is true.

ORA driver support for other recent Oracle features

Client result cache

The PowerBuilder ORA driver supports Oracle Client Cache, however this feature depends on your Oracle Server and Client configuration. You can configure the Oracle Client Cache with an *init.ora* or *sqlnet.ora* file. Cached queries are annotated with "`/*+ result_cache */`" hints to indicate that results are stored in the query result cache. To cache the client result set, you must also enable OCI statement caching from PowerBuilder applications with the StatementCache DBPARM parameter.

Application driver name

An OCI application can choose its own name and set it as a diagnostic aid. The AppDriverName DPBARM parameter allows you to set your own client driver name for the PowerBuilder ORA interface. The maximum length of the name is 8 characters. You can display the client driver name with the V$SESSION_CONNECT_INFO or GV$SESSION_CONNECT_INFO dynamic performance view queries.

Client access through a proxy

The PowerBuilder ORA driver supports the proxy authentication feature that was introduced in Oracle 10.2. With proxy authentication, the end user typically authenticates to a middle tier (such as a firewall), that in turn logs into the database as a proxy user. After logging into the database, the proxy user can switch to the end user's identity and perform operations using the authorization accorded to that user.

The ConnectAs DBParm parameter allows you to take advantage of this proxy connection feature. For example, if the user's Transaction object LogID is "Scott" and you set the ConnectAs DBParm parameter to "John", the OCI client logs in to database as the proxy user ("Scott"), then switches to the end user identity ("John").

If you are using connection or session pooling, the proxy user name is the connection or session pooling creator (which you can provide in the PoolCreator and PoolPwd DBParm parameters), and the Transaction object's LogID is ignored. No proxy session can be created if pooling is set to homogeneous session mode.

Limitation on proxy connection without pooling
When using a proxy connection without pooling, you must set the NLS_Charset DBPARM to "Local" or to another non-Unicode character set. If you do not change the "Unicode" default value for this DBPARM, the connection fails because the Oracle Client Interface does not accept a Unicode name string for its proxy client attribute.

Support for XMLType datatype

The PowerBuilder ORA driver supports the Oracle XMLType datatype that was introduced with Oracle 9*i*. The XMLType datatype is mapped to the PowerBuilder String datatype. However, you cannot use this datatype:

- In the Where clause of a PowerBuilder embedded SQL statement or in a DataWindow object
- As a parameter of a procedure or function, because PowerBuilder binds XMLType as a String datatype but Oracle does not support this usage
- In columns that you select directly in an Oracle cursor statement

  For example, if the col1 column has an XML datatype, you cannot select this column directly in an Oracle cursor as the following code attempts to do:

  ```
  CREATE OR REPLACE Function p_Ora_sp_char_11 return
    types.cursortype
  AS
  l_cursor types.cursorType;
  begin
  open l_cursor for select col1 from t_Ora_sp_char_11;
  return l_cursor;
  end;
  ```

  However, you can use the following code in a PowerBuilder application to obtain string values for the XMLType column that you select in an Oracle cursor statement:

  ```
  CREATE OR REPLACE Function p_Ora_sp_char_11 return
    types.cursortype
  AS
  l_cursor types.cursorType;
  begin
  open l_cursor for select x.col1.getstringval() from
      t_Ora_sp_char_11 x;
  return l_cursor;
  end;
  ```

Supported Oracle features not related to the ORA driver

PowerBuilder 11.5 applications can support the Oracle features listed in the following table, although these features are not enabled by DPBARM parameters or the ORA driver.

Supported feature	Description
Database Resident Connection Pooling (DRCP)	Provides a connection pool in the database server for typical Web application usage (where the application acquires a database connection, works on it for a relatively short time, and then releases it). Any application can leverage this pool by specifying: POOLED in the EZ Connect string or (SERVER=POOLED) in the TNS connect string. Before using this feature, the Oracle DBA must configure and start the DRCP.
Fault Diagnosability in OCI	This infrastructure helps diagnose problems such as those caused by data or metadata corruption, or by database code bugs. Oracle stores incident numbers and diagnostic information outside of the database in the Automated Diagnostic Repository (ADR).
Oracle Call Interface (OCI) Security Enhancement	Prevents disclosure of the database version string before authentication and posts warnings in the event of unauthorized access or when user actions are audited.

Database profile dialog box changes for the ORA driver

Pooling parameters

The database profile dialog box for an Oracle 11*g* connection includes a Pooling tab that lets you select the following pooling parameters:

Pooling parameter	Description
Pooling Type	You can select Session Pooling, Connection Pooling, or None (default). Sets the Pooling DBParm.
Runtime Connection Load Balancing	This check box is selected by default. It is ignored when you select Connection Pooling or None for the Pooling Type. Sets the RTConnBalancing DBParm.

Pooling parameter	Description
Homogeneous Session	This check box is not selected by default and is valid for session pooling only. If selected, all sessions in the pool are authenticated with the PoolCreator and PoolPwd DBParm parameters when these are provided. The user name and password in later connection requests are ignored. Proxy sessions cannot be created in homogeneous session mode. Sets the SessionHomogeneous DBParm.
Minimum Number of Sessions	Integer for the minimum number of database connection sessions; value is 1 by default. Sets the CSMin DBParm. This value is ignored when the SessionHomogeneous DBParm is set to false.
Maximum Number of Sessions	Integer for the maximum number of database connection sessions; value is 100 by default. Sets the CSMax DBParm.
Increment	Integer for database connection increments per session; value is 1 by default. Sets the CSIncr DBParm. This value is ignored when the SessionHomogeneous DBParm is set to false.
Pool Creator	User name used to create the connection or session pool when the pool is not already created. Sets the PoolCreator DBParm to a string for the user name prior to the database connection. If you do not provide a value for the PoolCreator DBParm, the Transaction object's LogID and LogPass properties are used to create the pooling.
Password	Password used to create the connection or session pool when the pool is not already created. Sets the PoolPwd DBParm to a string for the password for the pool creator.

Additional parameters for Oracle 11g support

Besides the pooling parameters, other database parameters provide additional support for Oracle 11*g* database connections. These are described in the following table:

Additional parameters for Oracle 11g support	Database profile setup tab page	Description
Application Driver Name (available for ORA driver only)	System	Allows you to set your own client driver name for diagnostic purposes. Sets the AppDriverName DPBARM.

Additional parameters for Oracle 11g support	Database profile setup tab page	Description
Number of Oracle Statements Cached (supported by the O10 and ORA drivers)	Transaction	The number of statements (0 by default) you can cache for each session. Must be set to a nonzero value to use with the Oracle Client Cache. Sets the StatementCache DBParm.
Connect As (supported by all available Oracle drivers, but allows entry of an end user name for the ORA driver only)	Connection	Editable drop-down list where you can enter an end user if you are using proxy authentication, or select the DEFAULT, SYSOPER, or SYSDBA user names. Sets the ConnectAs DBParm. You cannot use connection or session pooling if you set the ConnectAs DBParm to SYSDBA or SYSOPER.

Native driver support for MS SQL Server 2008

PowerBuilder support for connections to SQL Server 2008 databases includes new database parameters as well as support for new SQL Server datatypes. To connect to SQL Server 2008 from PowerBuilder, you must install the SNC 10.0 driver.

For more detailed information, see:

- New database parameters
- Support for new datatypes in SQL Server 2008
- T-SQL enhancements
- Unsupported SQL Server 2008 features

New database parameters

Provider parameter

The Provider DBParm parameter for the Microsoft SQL Native Client (SNC) interface allows you to select the SNC version that you want to use for a database connection. You can set this parameter in script to SQLNCLI (for the SNC 9.0 driver that connect to SQL Server 2005) or to SQLNCLI10 (for the SNC 10.0 driver that connects to SQL Server 2008). Otherwise, you can select one of these providers on the Connection tab of the Database Profile Setup dialog box for the SNC interface.

If you do not set or select a provider, the default selection is SQLNCLI (SNC 9.0 for SQL Server 2005). This allows existing SNC interface users to be able to migrate to PowerBuilder 11.5 without any modifications. If PowerBuilder fails to connect with the SQLNCLI provider, it will attempt to connect to SQLNCLI10 provider. However, if you explicitly set the provider and the connection fails, PowerBuilder displays an error message.

Failover parameter

The FailoverPartner DBParm parameter allows you to set the name of a mirror server, thereby maintaining database availability if a failover event occurs. You can also set the name of the mirror server on the System tab of the Database Profile Setup dialog box for the SNC interface.

When failover occurs, the existing PowerBuilder connection to SQL Server is lost. The SNC driver releases the existing connection and tries to reopen it. If reconnection succeeds, PowerBuilder triggers the DBNotification event.

The following conditions must be satisfied for PowerBuilder to trigger the failover event:

- The FailoverPartner DBParm is supplied at connect time
- The SQL Server database is configured for mirroring
- PowerBuilder is able to reconnect successfully when the existing connection is lost

When failover occurs:

- PowerBuilder returns an error code (998) and triggers the DBNotification event with notification type DBFailover!
- Existing cursors cannot be used and should be closed
- Any failed database operation can be tried again
- Any uncommitted transaction is lost. New transactions must be started

Support for new datatypes in SQL Server 2008

Date and time datatypes

The following table lists new SQL Server 2008 date and time datatypes and the PowerScript datatypes that they map to:

SQL Server datatype	PowerScript datatype
DATE	Date
TIME	Time (Supports only up to 6 fractional seconds precision although SQL Server datatype supports up to 7 fractional seconds precision.)
DATETIME2	DateTime (Supports only up to 6 fractional seconds precision although SQL Server datatype supports up to 7 fractional seconds precision.)

The SQL Server 2008 DATETIMEOFFSET datatype is not supported in PowerBuilder 11.5.

Precision settings When you map to a table column in a SQL Server 2008 database, PowerBuilder includes a column labeled "Dec" in the Column Specifications view of the DataWindow painter, and a text box labeled "Fractional Seconds Precision" in the Column (Object Details) view of the Database painter. These fields allow you to list the precision that you want for the TIME and DATETIME2 columns.

The precision setting is for table creation only. When retrieving or updating the data in a column, PowerBuilder uses only up to six decimal places precision for fractional seconds, even if you enter a higher precision value for the column.

Filestream datatype

The FILESTREAM datatype allows large binary data to be stored directly in an NTFS file system. Transac

statements can insert, update, query, search, and back up FILESTREAM data.

The SQL Server Database Engine implements FILESTREAM as a Varbinary(max) datatype. The PowerBuilder SNC interface maps the Varbinary(max) datatype to a BLOB datatype, so to retrieve or update filestream data, use the SelectBlob or UpdateBlob SQL statements, respectively. To specify that a column should store data on the file system, you must include the FILESTREAM attribute in the Varbinary(max) column definition. For example:

```
CREATE TABLE FSTest (
  GuidCol1 uniqueidentifier ROWGUIDCOL NOT NULL
  UNIQUE DEFAULT NEWID(),
  IntCol2 int,
  varbinaryCol3 varbinary(max) FILESTREAM);
```

Do not use PowerScript file access functions with FILESTREAM data
You can access FILESTREAM data by declaring and using the Win32 API functions directly in PowerBuilder applications. However, existing PowerBuilder file access functions cannot be used to access FILESTREAM files. For more information about accessing FILESTREAM data using Win32 APIs, see the MSDN SQL Server Developer Center Web site at http://msdn.microsoft.com/en-us/library/bb933877(SQL.100).aspx.

Using CLR datatypes in PowerBuilder

The binary values of the .NET Common Language Runtime (CLR) datatypes can be retrieved from a SQL Server database as blobs that you could use in PowerBuilder applications to update other columns in the database. If their return values are compatible with PowerBuilder datatypes, you can use CLR datatype methods in PowerScript, dynamic SQL, embedded SQL or in DataWindow objects, because the SQL script is executed on the SQL Server side.

The CLR datatypes can also be mapped to Strings in PowerScript, but the retrieved data is a hexadecimal string representation of binary data.

You can use the above methods to work with all datatypes that are implemented as CLR datatypes, such as the HierarchyID datatype, Spatial datatypes, and User-defined types.

HierarchyID datatype

HierarchyID is a variable length, system datatype that can store values representing nodes in a hierarchical tree, such as an organizational structure. A value of this datatype represents a position in the tree hierarchy.

ISQL Usage You can use HierarchyID columns with CREATE TABLE, SELECT, UPDATE, INSERT, and DELETE statements in the ISQL painter. For example:

```
CREATE TABLE Emp (
  EmpId int NOT NULL,
  EmpName varchar(20) NOT NULL,
  EmpNode hierarchyid NULL);
```

To insert HierarchyID data, you can use the canonical string representation of HierarchyID or any of the methods associated with the HierarchyID datatype as shown below.

```
INSERT into Emp VALUES (1, 'Scott',
  hierarchyid::GetRoot());
INSERT into Emp VALUES (2, 'Tom' , '/1/');

DECLARE @Manager hierarchyid
SELECT @Manager = hierarchyid::GetRoot() FROM Emp
```

```
INSERT into Emp VALUES (2, 'Tom',
  @Manager.GetDescendant(NULL,NULL));
DECLARE @Employee hierarchyid
SELECT @Employee = CAST('/1/2/3/4/' AS hierarchyid)
INSERT into Emp VALUES (2, 'Jim' , @Employee);
```

You cannot select the HierarchyID column directly since it has binary data, and the ISQL painter Results view does not display binary columns. However, you can retrieve the HierarchyID data as a string value using the ToString method of HierarchyID. For example:

```
Select EmpId, EmpName, EmpNode.ToString() from Emp;
```

You can also use the following methods on HierarchyID columns to retrieve its data: GetAncestor, GetDescendant, GetLevel, GetRoot, IsDescendant, Parse, and Reparent. If one of these methods returns a HierarchyID node, then use ToString to convert the data to a string. For example:

```
Select EmpId, EmpName, EmpNode.GetLevel() from Emp;
Select EmpId, EmpName,
   EmpNode.GetAncestor(1).ToString() from Emp;
```

HierarchyID columns can be updated using a String value or a HierarchyID variable:

```
Update Emp Set EmpNode = '/1/2/' where EmpId=4;
Delete from Emp where EmpNode = '/1/2/';
```

PowerScript Usage You can use HierarchyID columns in embedded SQL statements for SELECT, INSERT, UPDATE, and DELETE operations. HierarchyID data can be retrieved either as a String or as a Binary(Blob) datatype using the SelectBlob statement.

When using a String datatype to retrieve HierarchyID data, use the ToString method. Otherwise the data will be a hexadecimal representation of the binary HierarchyID value.

The following example shows how you can use HierarchyID methods in embedded SQL:

```
long id
String hid,name
Select EmpId, EmpName, EmpNode.ToString()
  into :id, :name, :hid
  from Emp where EmpId=3;
Select EmpId, EmpName, EmpNode.GetLevel()
  into :id, :name, :hid
  from Emp where EmpId=3;
```

```
Blob b
Selectblob EmpNode into :b from Emp where EmpId =2;
```

DataWindow Usage DataWindow objects do not directly support the HierarchyID datatype. But you can convert the HierarchyID to a string using the ToString method or an associated HierarchyID method in the data source SQL. For example:

```
SELECT EmpId, EmpName, EmpNode.ToString() FROM Emp;
SELECT EmpId, EmpName, EmpNode.GetLevel() FROM Emp;
```

Spatial datatypes

Microsoft SQL Server 2008 supports two spatial datatypes: the geometry datatype and the geography datatype. In SQL Server, these datatypes are implemented as .NET Common Language Runtime (CLR) datatypes.

Although the PowerBuilder SNC interface does not work with CLR datatypes, you can convert the spatial datatypes into strings (with the ToString function) and use them in PowerScript, in the ISQL painter, in embedded SQL, and in DataWindow objects. This is similar to the way you use the HierarchyID datatype. The SelectBlob SQL statement also lets you retrieve binary values for these datatypes.

The geography and geometry datatypes support eleven different data objects, or instance types, but only seven of these types are instantiable: Points, LineStrings, Polygons, and the objects in an instantiable GeometryCollection (MultiPoints, MultiLineStrings, and MultiPolygons). You can create and work with these objects in a database, calling methods associated with them, such as STAsText, STArea, STGeometryType, and so on. For example:

```
CREATE TABLE SpatialTable (id int IDENTITY (1,1),
  GeomCol geometry);
INSERT INTO SpatialTable (GeomCol) VALUES (
  geometry::STGeomFromText(
   'LINESTRING (100 100,20 180,180 180)',0));
select id, GeomCol.ToString() from SpatialTable;
select id, GeomCol.STAsText(),
  GeomCol.STGeometryType(),
  GeomCol.STArea() from SpatialTable;
```

User-defined types

User-defined types (UDTs) are implemented in SQL Server as CLR types and integrated with .NET. Microsoft SQL Server 2008 eliminates the 8 KB limit for UDTs, enabling the size of UDT data to expand dramatically.

Although the PowerBuilder SNC interface does not directly support UDT datatypes, you can use the ToString method to retrieve data for UDTs in the same way as for other CLR datatypes such as HierarchyId or the spatial datatypes. However, if a UDT datatype is mapped to a String datatype in PowerScript, UDT binary values will be retrieved as hexadecimal strings. To retrieve or update data in binary form (blob) from a UDT, you can use the SelectBlob or UpdateBlob SQL statements, respectively.

You can use any of the associated methods of UDT or CLR datatypes that return compatible data (such as String, Long, Decimal, and so on) for PowerBuilder applications.

T-SQL enhancements

MERGE statement

The MERGE Transact-SQL statement performs INSERT, UPDATE, or DELETE operations on a target table or view based on the results of a join with a source table. You can use MERGE statement in the ISQL painter and in PowerScript using dynamic SQL. For example

```
String mySQL
mySQL = "MERGE INTO a USING b ON a.keycol = b.keycol " &
  + "WHEN MATCHED THEN "&
  + "UPDATE SET col1 = b.col1,col2 = b.col2 " &
  + "WHEN NOT MATCHED THEN " &
  + "INSERT (keycol, col1, col2, col3)" &
  + "VALUES (b.keycol, b.col1, b.col2, b.col3) " &
  + "WHEN SOURCE NOT MATCHED THEN " &
  + "DELETE;"
EXECUTE IMMEDIATE :Mysql;
```

Using the MERGE statement in ISQL
A MERGE statement must be terminated by a semicolon. By default the ISQL painter uses a semicolon as a SQL terminating character, so to use a MERGE statement in ISQL, the terminating character must be changed to a colon (:), a forward slash (/), or some other special character.

Grouping sets

GROUPING SETS is an extension of the GROUP BY clause that lets you define multiple groupings in the same query. GROUPING SETS produce a single result set, making aggregate querying and reporting easier and faster. It is equivalent to a UNION ALL operation for differently grouped rows.

The GROUPING SETS, ROLLUP, and CUBE operators are added to the GROUP BY clause. A new function, GROUPING_ID, returns more grouping-level information than the existing GROUPING function. (The WITH ROLLUP, WITH CUBE, and ALL syntax is not ISO compliant and is therefore deprecated.)

The following example uses the GROUPING SETS operator and the GROUPING_ID function:

```
SELECT EmpId, Month, Yr, SUM(Sales) AS Sales
  FROM Sales
  GROUP BY GROUPING SETS((EmpId, ROLLUP(Yr, Month)));
SELECT COL1, COL2,
  SUM(COL3) AS TOTAL_VAL,
  GROUPING(COL1) AS C1,
  GROUPING(COL2) AS C2,
  GROUPING_ID(COL1, COL2) AS GRP_ID_VALUE
  FROM TEST_TBL GROUP BY ROLLUP (COL1, COL2);
```

You can use the GROUPING SETS operator in the ISQL painter, in PowerScript (embedded SQL and dynamic SQL) and in DataWindow objects (syntax mode).

Row constructors

Transact-SQL now allows multiple value inserts within a single INSERT statement. You can use the enhanced INSERT statement in the ISQL painter and in PowerScript (embedded SQL and dynamic SQL). For example:

```
INSERT INTO Employees VALUES ('tom', 25, 5),
  ('jerry', 30, 6), ('bok', 25, 3);
```

When including multiple values in a single INSERT statement with host variables, you must set the DisableBind DBParm to 1. If you use literal values as in the above example, you can insert multiple rows in a single INSERT statement regardless of the binding setting.

Compatibility level

In SQL Server 2008, the ALTER DATABASE statement allows you to set the database compatibility level (SQL Server version), replacing the sp_dbcmptlevel procedure. You can use this syntax in the ISQL painter and in PowerScript (dynamic SQL). For example:

```
ALTER DATABASE <database_name>
  SET COMPATIBILITY_LEVEL = {80 | 90 | 100}
80 = SQL Server 2000
90 = SQL Server 2005
100 = SQL Server 2008
```

Compatibility level affects behaviors for the specified database only, not for the entire database server. It provides only partial backward compatibility with earlier versions of SQL Server. You can use the database compatibility level as an interim migration aid to work around differences in the behaviors of different versions of the database.

Table hints

The FORCESEEK table hint overrides the default behavior of the query optimizer. It provides advanced performance tuning options, instructing the query optimizer to use an index seek operation as the only access path to the data in the table or view that is referenced by the query. You can use the FORCESEEK table hint in the ISQL painter, in PowerScript (embedded SQL and dynamic SQL), and in DataWindow objects (syntax mode). For example:

```
Select ProductID, OrderQty from SalesOrderDetail
   with (FORCESEEK);
```

Unsupported SQL Server 2008 features

The PowerBuilder SNC interface does not support the User-Defined Table Type (a user-defined type that represents the definition of a table structure) that was introduced in SQL Server 2008.

Transaction object enhancement

PowerBuilder 11.5 adds two new events for the Transaction object, DBError and SQLPreview. These events have slightly different signatures than the DBError and SQLPreview events on the DataWindow control and DataStore, but their behavior is similar.

For event signature information, see the online Help for the Transaction object and DataWindow control DBError and SQLPreview events.

Use with embedded SQL

By default, whenever an error occurs in the Transaction object, the DBError event is called. The error code and error message are passed to this event. You can add code to the DBError event to handle these errors. The SQLPreview event on the Transaction object is triggered before SQL statements are passed to the DBMS.

Use with DataWindow or DataStore

DBError When using a Transaction object with a DataWindow or DataStore, the DataWindow DBError event is triggered before the DBError event of the Transaction object. To coordinate these two events, two additional return values have been added for the DataWindow DBError event, and the meaning of the existing return values is expanded to indicate whether the transaction object's DBError event should be fired.

The meaning of the existing return values for the DataWindow DBError event are modified as follows:

0	Display the error message and trigger the Transaction object's DBError event if it is defined.
1	Do not display the error message, and trigger the Transaction object's DBError event if it is defined.

In PowerBuilder 11.5, two new return values have been added for the DBError event of the DataWindow and DataStore:

2	Display the error message and ignore the Transaction object's DBError event whether it is defined or not.
3	Do not display the error message and ignore the Transaction object's DBError event whether it is defined or not.

SQLPreview When using the Transaction object with a DataWindow or DataStore, the DataWindow or DataStore SQLPreview event is triggered first. If the return value of the DataWindow or DataStore’s SQLPreview event is 0 (Continue processing), the Transaction object's SQLPreview event is triggered.

FDCC compliance

About FDCC

The Federal Desktop Core Configuration (FDCC) is a security standard mandated by the US Office of Management and Budget (OMB). To meet the FDCC security requirements, PowerBuilder 11.5 can be installed only by a system administrator. However, PowerBuilder and the applications that you develop with PowerBuilder are designed to be run in a standard user context without elevated system administration privileges.

Although most PowerBuilder files install by default to *Program Files\Sybase* subdirectories, write access to these subdirectories is typically restricted to administrative users. Therefore, to meet the FDCC requirements, all writable files are installed, copied, or created in directories where standard users have write access.

FDCC constraints on certain PowerBuilder features

Several PowerBuilder features might still require write access to *Program Files\Sybase* subdirectories, or require the ability to add a system printer. For example, to save a DataWindow to a PDF file, you must first copy the *PSCRIPT.DLL*, *PSCRIPT.NTF*, and *PS5UI.DLL* files to the *Program Files\Sybase\Shared\PowerBuilder\drivers* directory, and you must install the Sybase® Datawindow PS printer. This must be done by an administrator before a standard user can save a DataWindow to a PDF file.

When using remote debugging on an FDCC-configured desktop with the Windows Firewall set to on, the connection to the server fails if *pb115.exe* is not included in the list of firewall exceptions at the domain level. To use remote debugging under these circumstances, you must add *pb115.exe* to (or select PowerBuilder 11.5 from the program list for) the list of domain-level firewall exceptions.

Files shared by all users

As part of its FDCC compliance configuration, PowerBuilder installs writable files that are shared by all users in the *C:\Documents and Settings\All Users\Documents\Sybase\PowerBuilder 11.5* directory on Windows XP and Windows 2003, and in *C:\Users\Public\Documents\Sybase\PowerBuilder 11.5* on Windows Vista and Windows 2008. These files include:

- The EASDemo databases (*easdemo115.db* and *easdemo115u.db*)
- All Code Examples directories and files
- The PowerBuilder Windows Help and compiled HTML Help files
- The Translation Toolkit directories and files

Files reserved for Individual users

Other writable files are installed in the default Program Files\Sybase subdirectories, but are copied to different locations the first time a user starts PowerBuilder. In this way, each PowerBuilder user gets a private copy of these files.

The following table lists the files that are copied and updated in the directories of all users who run an instance of PowerBuilder. The path variable in the table header (*UserName*) stands for the user name of a PowerBuilder user. For Windows XP and 2003, this is under the *C:\Documents and Settings* directory. For Windows Vista and 2008, this is under the *C:\Users* directory.

In C:\...\UserName\ subdirectory	Files copied or updated
On Windows XP and 2003: *Local Settings\Application Data\Sybase\PowerBuilder 11.5* On Windows Vista and 2008: *AppData\Local\Sybase\PowerBuilder 11.5*	• Initialization files (*PB.INI*, *PBLAB115.INI*, *PBODB115.INI*) • License files (*PB115.LIC*, *pb115_sysam.properties*)
On Windows XP and 2003: *My Documents\Sybase\PowerBuilder 11.5\Tutorial* On Windows Vista and 2008: *Documents\Sybase\PowerBuilder 11.5\Tutorial*	• Files for the PowerBuilder Getting Started tutorial

The locations of writable PowerBuilder files reserved for individual use are set in HKEY_CURRENT_USER registry entries for each PowerBuilder user. For example, the location of the *PB.INI* file that is copied to each user's local application data directory is registered under the registry key *HKEY_CURRENT_USER\Sybase\PowerBuilder\11.5\InitPath*.

Silent install and uninstall

You can install and uninstall PowerBuilder 11.5 without displaying messages or windows during the setup or removal process. However, you must accept the Sybase license agreement before you can run the silent install file. You can indicate your acceptance by typing the following line in a DOS command box before you type in the silent install batch file command:

```
SET  AgreeToSybaseLicense=true
```

If the DOS prompt does not display the main PowerBuilder installation directory, you must change to that directory. From the same DOS command box, you can then run the silent install using the following syntax:

silentinstall.bat [“lic=*licPathNameOrServerName*” “opt=*product*” “dir=*directoryName*” “shr=*sharedDirectoryName*” “log=*logFileName*”]

Modifying the batch file directly
You can indicate your acceptance of the Sybase license agreement by directly editing the *silentinstall.bat* file. You do this by changing the line in the batch file that reads `set AgreeToSybaseLicense=false`. You must change this line to read `set AgreeToSybaseLicense=true`. After you make this change, you can double-click the *silentinstall.bat* file to run the install with default selections rather than running it from a command line.

You can use the DOS command line to provide help for silent install parameters. The help is available by typing `Help` or a question mark after entering `silentinstall.bat` on the command line.

All of the parameters for the silent install are optional. If you do not provide a value for the “lic” parameter, the installer searches the files in the installer directory in ascending alphabetic order for a valid license file. It uses the first valid license file it finds to install PowerBuilder and InfoMaker®. If you do not provide the “lic” parameter and there is no valid license file in the installer search path, an evaluation version of PowerBuilder is installed.

The parameters can be listed in any order and are not case sensitive. Parameter values are also not case sensitive.

Silent install parameter	Description
licPathNameOrServerName	The full path to a valid license file with an LIC extension, or the name of a server hosting a served license. If a license server requires a port number, you can include the port number after the server name separated by a colon. For example: silentinstall.bat “lic=myServer:1688”
product	Names the product or products to install. Values are: • **All** (default) Installs PowerBuilder and InfoMaker • **PB** Installs PowerBuilder only • **IM** Installs InfoMaker only
directoryName	Indicates the main installation directory. If you omit this parameter, the main product components install to the *sysDriver*\Program Files\Sybase\productName directory, where *sysDriver* is the main computer drive and *productName* is either PowerBuilder 11.5 or InfoMaker 11.5.
sharedDirectoryName	Indicates the shared directory for PowerBuilder or InfoMaker. If you omit this parameter, this directory installs to *sysDriver*\Program Files\Sybase\Shared.
logFileName	Names the log file for the installation. If you omit this parameter, the log file is written to the system *Temp* directory with the file name *silentinstall.log*.

If the license file you point to in the *licPathNameOrServerName* parameter is a professional or desktop license, InfoMaker will not be installed.

The following example uses all five parameters for the silent install command:

```
silentinstall.bat "lic=D:\pb115\pb115.lic"
```

```
"opt=pb" "dir=D:\pb115" "shr=D:\shared"
"log=D:\install.log"
```

When installing PowerBuilder, the silent install writes warnings to its log file if the setup computer does not have any of the following items: SQL Anywhere® 11, Microsoft .NET Framework 2.0 or later, Microsoft .NET Framework 2.0 SDK or later, Microsoft IIS 5.0 or later, or AJAX Extensions version 1.0. When installing InfoMaker, the silent install includes a warning in the log file if the setup computer does not have a working copy of SQL Anywere 11.0. If SQL Anywhere 11.0 is not found, the demonstration databases and code examples are not installed for PowerBuilder or for InfoMaker.

You run the standard silent uninstall by double-clicking the *silentuninstall.bat* file or running it from a command line.

Feature deprecation

The following items have been removed and are no longer available in PowerBuilder starting with the 11.5 release:

- IE Web Control (PBWebControlSource) option in Web Forms projects
- Oracle 8/8i (O84) database interface
- JSP targets (use Sybase Workspace instead)
- Automation Server projects
- DataWindow plug-in and PowerBuilder window plug-in
- PowerBuilder Window ActiveX
- UDDI browser in Web Service Proxy projects