Upgrading PowerBuilder Applications

PowerBuilder® 2021

FOR WINDOWS

DOCUMENT ID: DC20251-01-1900-01

LAST REVISED: October 22, 2021

Appeon Inc., 1/F, Shell Industrial Building, 12 Lee Chung Street, Chai Wan District, Hong Kong.

Contents

1 Purpose

This document addresses changes between PowerBuilder 6.5 and PowerBuilder 2021. Read all sections that apply to your application. Subsections with IM in parentheses after their titles also apply to InfoMaker.

2 Overview

You can upgrade a PowerBuilder application from any version of PowerBuilder directly to any later version. Before you upgrade to a later version, read this document to learn about changes in PowerBuilder that might affect your application.

3 Upgrading from any PowerBuilder version (including beta)

3.1 Migration Assistant

Before opening PBLs that were created in an earlier version, use the Migration Assistant in the new version to check them for obsolete syntax or the use of new reserved words. To open the Migration Assistant, select File>New from the PowerBuilder menu bar and select Migration Assistant from the Tools page of the New dialog box.

3.2 Upgrading PBLs and performing a full build (IM)

You must upgrade PBLs created with earlier versions of PowerBuilder to the new version. You should always back up PBLs and PBTs before you upgrade.

Make sure the database connection is successful in the new version of PowerBuilder IDE before you upgrade, as the database schema is required in order to properly migrate the DataWindow objects and SQLs.

The Migrate Application dialog box opens automatically after you open a workspace that contains PowerScript targets built using an earlier version. If you add a PBT containing such PBLs to an open workspace, or add PBLs built in an earlier version of PowerBuilder to a target's library list, the Migrate Application dialog box does not open automatically. To open the dialog box, select each target that contains PBLs created using earlier versions of PowerBuilder in the System Tree and select Migrate from the pop-up menu.

Perform a full build to PBLs after they are upgraded successfully.

Please read Section 5.9, “Upgrading targets” in *Users Guide* before upgrading your applications.

3.3 Upgrading database profiles (IM)

To use database profiles that were set up in an earlier version of PowerBuilder, right-click any item in the Database Profiles dialog box in the earlier version of PowerBuilder and select Export Profile(s) from the pop-up menu. You can then import the profiles you select in the Database profiles dialog box in the later version of PowerBuilder.

Note that 1) the driver "SYJ Sybase ASE for EAServer" is removed from PowerBuilder 2017 and later versions, therefore, database profiles created on this driver will not be imported; 2) the EAServer tab page is removed from the database profile settings for JDB JDBC/O10 Oracle 10g/O90 Oracle 9i/ODB ODBC/ORA Oracle, so the corresponding settings will not be imported to PowerBuilder 2017 and later versions.

3.4 Runtime DLLs (IM)

The applications that you build using any version of PowerBuilder should be deployed with the PowerBuilder runtime DLLs from the same version. If the development computer is updated with a new build, PowerBuilder applications and components *must* be rebuilt and redeployed with the new runtime files.

3.5 Upgrading PowerBuilder Extension applications

Import the PowerBuilder extension file (pbXXX###.pbx, where XXX is an abbreviation used for the extension type, and ### stands for the current PowerBuilder version) into your upgraded PowerBuilder project, then do a full build before deploying the application.

3.6 GetFileOpenName Initdir parameter behavior

The initdir parameter in GetFileOpenName uses some different algorithms in Windows 7/8.1/10 than in Windows 2000/XP/Vista.

3.6.1 Windows 7/8.1/10:

1. If initdir has the same value that was selected in the first instance of the application's Open or Save As dialog box, then it uses the path the user selected most recently as the initial directory.
2. If filename contains a path, that path is the initial directory.
3. If initdir is not NULL, it specifies the initial directory.
4. If initdir is NULL and the current directory contains any files of the specified filter types, the initial directory is the current directory.
5. Otherwise, the initial directory is the personal files directory of the current user.
6. If no other conditions are met, the initial directory is the Desktop folder.

3.6.2 Windows 2000/Windows XP/Windows Vista:

1. If filename contains a path, that path is the initial directory.
2. Otherwise, initdir specifies the initial directory
3. If the application has used an Open or Save As dialog box in the past, the path most recently used is selected as the initial directory. However, if an application is not run for a long time, its saved path is discarded.
4. If initdir is NULL and the current directory contains any files of the specified filter types, the initial directory is the current directory.

4 Upgrading from PowerBuilder 2019 R2 or earlier

4.1 Class name and location changes

The separation of PowerBuilder Runtime from PowerBuilder IDE (starting from version 2019 R3) brings in the following major changes which may causes breaks in the application source code:

- **ALL** runtime files, including .dll, .ini, .pbx, .pbd etc. are renamed so that the version number indicator (such as "170", "190") that used to be appended to the file name is removed, for example, pbvm190.dll is renamed as pbvm.dll, pbodb190.ini is renamed as pbodb.ini, pbwsclient190.pbd is renamed as pbwsclient.pbd, pborc190.dll is renamed as pborc.dll.

  Please review your application source code carefully and update the file name and path accordingly, especially

  - If your application uses pbwsclient.pbd/pbwsclient.pbx, pbdom.pbd/pbdom.pbx, pbsoapclient.pbd/pbsoapclient.pbx, pbejbclient.pbd/pbejbclient.pbx
  - If your application uses pborc.dll to import/export source from PBLs
  - If your applications calls pbvm.dll

- **ALL** runtime files that used to be installed to the "Shared" folder are now installed to the "Runtime [version]" folder and the "IDE" folder separately.

- The runtime file location is no longer recorded in the PATH system environment; it is recorded in the system registry instead.

If your application has called the runtime file, you may need to change the code accordingly to locate the updated file name in the new location. For code examples, refer to here.

4.2 Upgrading PowerBuilder projects

When you open a 2019/2019 R2 workspace in the 2019 R3 IDE, make sure to manually perform a Migrate and a Full Build to the source code; even though you get no prompts from IDE for Migrate and Full Build.

When you upgrade a target which uses the UI theme feature, from 2019/2019 R2 to 2019 R3, make sure to update the default theme path and use the theme files of 2019 R3: 1) if you have specified to use the default theme path in the target, modify the theme path in the Application properties dialog box (from "%AppeonInstallPath%\Shared\PowerBuilder \theme190" to "%AppeonInstallPath%\PowerBuilder 19.0\IDE\theme"); 2) after creating the application executable file, copy the "theme" folder from "%AppeonInstallPath% \PowerBuilder 19.0\IDE" to the same folder as the application executable file; the folder name "theme" cannot be changed, and the "theme190" folder is not effective any more. For detailed instructions, refer to the section called “Applying a theme” in Users Guide.

5 Upgrading from PowerBuilder 2017 or earlier

5.1 RichTextEdit control and RichText DataWindow changes

Starting from version 2019, PowerBuilder incorporates a special OEM version of TX Text Control ActiveX as the recommended rich text editor for the RichTextEdit control and the RichText DataWindow presentation style. TX Text Control ActiveX used to be the rich text editor in PowerBuilder 12.6 and earlier versions. The PowerBuilder developer can use this editor for free. The InfoMaker developer is not offered with this option, they can only use the TE Edit Control added in PowerBuilder 2017 or purchase TX Text Control ActiveX.

5.2 Upgrading Web service proxy

Creating Web service proxy for connecting to SOAP server is no longer eligible for technical support. As a replacement, developers who build Web service client that connects to SOAP server can use HTTPClient to call SOAP Web service. For more information, refer to this article: Call SOAP Web Service Using HTTPClient Object.

5.3 Migration path containing double-byte characters

To upgrade an application which is located in the path that contains double-byte characters (such as Chinese, Japanese, Korean etc.), you must first change the path to contain only single-byte characters, and then upgrade the application to the current PowerBuilder. Otherwise, you may encounter the error "Could not add target because of a bad application library in ...".

5.4 Connecting to database before upgrade

Before upgrading an application which contains DataWindow and E-SQL, you should connect to the database in the Database Painter first. Otherwise, you may encounter the error "C0042: Cannot compile without valid database sign-on."

5.5 Registry functions

Starting from PowerBuilder 2019, the registry functions (including RegistryDelete, RegistryKeys, RegistryGet, RegistrySet, & RegistryValues) are no longer automatically redirected to the Wow6432Node registry entry in a 64-bit operating system; these functions will directly access the 64-bit registry entries.

6 Upgrading from PowerBuilder 12.6 or earlier

6.1 Oracle JDK replaces SAP JVM (IM)

PowerBuilder 2017 and later do not use the SAP JVM; it uses Oracle JDK 1.6.

6.2 Appeon branding

All the company branding mentioned in the product and the documentation is updated to Appeon for InfoMaker and PowerBuilder.

For current information on PowerBuilder and InfoMaker documentation, go to: https://docs.appeon.com.

For latest release notes on PowerBuilder and InfoMaker, go to https://docs.appeon.com/pb2019r3/release_bulletin_for_pb/index.html (for PowerBuilder) and https://docs.appeon.com/im2019r3/release_bulletin_for_im/index.html (for InfoMaker).

6.3 "PowerBuilder Classic" renamed (IM)

"Classic" is removed from "PowerBuilder Classic", therefore, any mentioning of "PowerBuilder" or "PowerBuilder IDE" in version 2017 and later (product and documentation) is equivalent to "PowerBuilder Classic" or "PowerBuilder Classic IDE" in the old version. The same applies to InfoMaker.

6.4 .NET assembly/Web service not available in Standard Edition

The .NET Assembly and the .NET Web Service are supported in the CloudPro Edition and the Professional Edition of PowerBuilder 2017 and later, but **not** in the Standard Edition.

6.5 RichTextEdit control and RichText DataWindow changes (IM)

PowerBuilder 2017 and later use a new rich text editor to support the RichTextEdit control and the RichText DataWindow presentation style. The new editor supports almost the same properties and functions of rich text objects in previous versions of PowerBuilder, except for a few differences. For example, for the new editor, the InputFieldBackColor property will not take effect until you save the data into a PDF file or print the data; which means, when you preview the UI in the design view or when you run the UI, you will always see the background color is gray, only when you save the data to a PDF file or print the data, you will see the background color is changed to what you set. For the new editor, the Wordwrap property is always true (even when it is set to false, it is treated as true). For more about the differences, see the *What's New* document in the PowerBuilder Help.

For the PowerBuilder developer, they have the option to use the new editor for free, or to continue using the old editor (the old editor needs to be purchased separately). For the InfoMaker developer, they can only use the new editor; they cannot use the old editor any more.

The new editor supports the RichTextEdit control, RichText DataWindow Object, and the RichText edit style column. The SP1, SP2, and SP3 of TX Text Control ActiveX 2400 do

not support the RichText edit style (You should use TX Text Control ActiveX 2400 for supporting all these three features).

6.6 PB .NET IDE and WPF Projects Removed

PowerBuilder .NET IDE and WPF projects are discontinued since 2017.

6.7 EAServer Projects/Targets Removed

PowerBuilder 2017 and later have removed the EAServer-related projects, targets, objects, drivers (SYJ Sybase ASE for EAServer) etc., because the EAServer application server is an obsolete product. Use the migration assistant to detect and remove the deprecated features, components, etc. You can consider to convert server-side NVOs to client-side NVOs and/or deploy server-side NVOs as .NET Web service to Microsoft IIS.

6.8 Windows Form Removed

PowerBuilder 2017 and later have removed the Windows Form-related projects, targets, tools, etc., because it no longer supports the Windows Form application.

6.9 Web DataWindow Removed

PowerBuilder 2017 and later have also removed the projects/targets/tools related with the DataWindow Web control for ActiveX and Web DataWindow, due to the obsolete technology.

7 Upgrading from PowerBuilder 12.5.2 or earlier

7.1 SAP JVM replaces JDK

PowerBuilder no longer uses the Java Runtime Environment (JRE) or the Java Development Kit (JDK). It uses the SAP JVM. Disregard mention of the JDK in the online help.

7.2 SAP branding

All the company branding mentioned in the documentation is updated to SAP for InfoMaker and PowerBuilder.

For technical support, go to https://support.sap.com/home.html.

All the company branding mentioned in the documentation is updated to SAP for InfoMaker and PowerBuilder.

For current information on PowerBuilder and InfoMaker documentation go to: http://help.sap.com/powerbuilder and http://help.sap.com/infomaker.

7.3 New SoapConnection functions

As of PowerBuilder 12.5.x, there are two functions in the SoapConnection class for the .NET Web services engine only:

- EnablePreAuthentication- Applies only to webservices that use basic authentication mechanism. Call this function to pre-authenticate the request. Syntax: <conn>.EnablePreAuthentication()
- SetKeepAlive- Whether to make a persistent connection to the Internet resource. Syntax: <conn>.SetKeepAlive(Boolean <value)

7.4 Upgrading .NET Targets from Earlier Versions of PowerBuilder

For .NET Web Service components, you might need to remove or install supporting files on the development and deployment computers.

For .NET projects, verify or complete the following steps before redeploying the upgraded .NET projects:

- For all .NET targets, uninstall earlier versions of the PowerBuilder runtime files (system assemblies and windows 32 dlls) on all deployment machines. Then install the runtime files for the current version of PowerBuilder using the Runtime Packager or another tool, as described in the "Checklist for deployment" section of the Deploying Applications and Components to .NET.
- For .NET Web Service targets, clear the ASP.NET temporary files for the application or component on the development and deployed machines. The temporary files are located in the C:\Windows\Microsoft.NET\Framework\<version>\Temporary ASP.NET Files

\<projectName> directory, where <version> is typically v2.0.50727, and <projectName> is the project's Web application name or its Web Service virtual directory name.

- For .NET Windows Forms projects, select *FULL* for the project rebuild type before you redeploy your applications from PowerBuilder Classic for the first time.
- To migrate .PBWX solutions from an earlier version, open its .PBWX file using the File menu. If you double-click or drag-and-drop the file into PowerBuilder .NET 12.6, the Visual Studio Conversion wizard opens, inappropriately. You can use the double-click and drag-and-drop methods to open PowerBuilder 12.6 solutions.

7.5 Upgrading EAServer Targets

PowerBuilder .NET does not support EAServer targets.

7.5.1 Accessing EAServer Components from .NET Targets

PowerBuilder installs the EAServer dlls, *com.sybase.iiop.net.dll* and *com.sybase.ejb.net.dll* in the Global Assembly Cache (GAC) on the development computer and requires these dlls on the runtime computer for applications that access EAServer components from .NET targets. Use the Runtime Packager to install these dlls in the GAC on the deployment computers.

NOTE: The dlls installed by the PowerBuilder setup program have been tested, and are compatible with EAServer 6.3.1. However, if you use an EAServer version other than 6.3.1, SAP recommends that you copy the (assembly) dlls from the EAServer/lib directory to the GAC of all computers where you deploy your applications. If you then run or re-run the Runtime Packager, you must also re-copy these (assembly) dlls to the GAC, because the Runtime Packager overwrites them.

7.6 ImportFile size limit

See section 13.3 note for an update.

7.7 OData Support

PowerBuilder Classic and PowerBuilder .NET can use OData datasources.

7.7.1 Database Profile

Define a database profile to access an OData service in PowerBuilder using the OData interface.

7.7.2 Database Painter

You can connect to the OData service and work with in the Database painter.

- Define an OData datasource with the database profile painter
- Connect to an OData datasource in the Objects view
- Add tables to the Object Layout view with drag-and-drop or the context menu

- Display table and column properties in the Properties view with drag-and-drop or the context menu
- Drag-and-drop a table onto the Columns View
- Invoke the Edit Data feature in the Results view of the Database Painter

Since OData is not a SQL source, some features are not supported:

- The ISQL session view is not enabled
- You cannot create, modify or delete tables
- You cannot create users or group of users
- You cannot embed SQL statements in powerscript

7.7.3 Create a DataWindow Using an OData Service

Select the OData Service datasource in the DataWindow wizard.

7.7.4 Set the Connection Information for the OData Service

Use the SQLCA Transaction object (or user-defined transaction object) to retrieve and display data from the OData service in a DataWindow or Datastore.

7.8 64-bit Windows Applications

Create 64-bit native application in PowerBuilder Classic. To build a 64-bit application, select the platform in the Project painter General tab.

If you need to deliver both 32-bit and 64-bit versions of your application, you should use separate projects and separate folders for the deploy output.

There is no IDE for 64-bit development. Design time uses the same 32-bit interface and 64-bit features display at runtime when you deploy the application. When you click the running man button, the project runs as a 32-bit application.

32-bit remains the default for new and upgraded applications. During the deploy process, PowerBuilder checks and reports unsupported features used in the application.

7.8.1 New Property for Environment Object

The new ProcessBitness property identifies whether the application is a 32-bit or 64-bit process.

- Datatype -- integer
- Values -- 32 stands for 32-bit, and 64 stands for 64-bit

See Section 2.29, “Environment object” in *Objects and Controls* for more about the Environment object. See Section 10.249, “GetEnvironment” in *PowerScript Reference* to read about the *GetEnvironment* function.

7.8.2 New Datatype

The longptr datatype is 4 bytes in the 32-bit platform and 8 bytes in the 64-bit platform. In the 32-bit platform, longptr is the same as long; you can continue using long where longptr is required in 32-bit applications. In 64-bit applications, using long to hold longptr variables will lead to data truncation from 8 bytes to 4 bytes, or memory corruption if you pass a long ref variable when a longptr ref is required. If you want to move to 64-bit, use longptr wherever required. It does no harm to 32-bit.

Since PowerBuilder does not have a datatype corresponding to the C++ pointer type, and there are no pointer operations in PowerBuilder, longptr is not full-fledged PowerBuilder datatype. You can use it to hold/pass window handles, database handles and other objects that re essentially memory addresses. Doing complex operations on longptr type might not work. If you want to represent/compute 8-byte long integers, use longlong.

7.8.3 System Requirements

The design time environment requires:

- Windows SDK for Windows 7 or later
- .NET Framework 4.0 or later
- 64-bit Windows OS to test (development requires 32-bit only)

The runtime environment requires:

- 64-bit Windows OS
- PowerBuilder 12.6 64-bit system files
- 64-bit third-party libraries, such as database drivers and external dlls
- Greater than 4GB physical memory to avoid performance issues

7.8.4 Limitations

- To consume Web services, you must use the .Net engine. EasySOAP is not supported
- You can use OLE and ActiveX components in your applications, but you must use the 32-bit versions in the PowerBuilder Classic IDE. At runtime you must have the correct 64-bit ActiveX components installed.
- The RichText DataWindow header does not display when the HeaderFooter property is true until you call ShowHeadFoot(true). If you do not:
  - Selecttext (long ll, long c1, longl2, long c2, band b Header!) returns 0 and selected text is "" (string with 0 length)
  - Selecttext (long ll, long c1, long l2, long c2, band b Footer!) returns 0 and selected text is "" (string with 0 length)

- Scrolling in a RichText DataWindow loses focus
- CopyRTF(false,header!) works only when you call ShowHeadFoot(true) when Header/Footer is true
- InsertDocument("*.htm", true) returns -1
- InsertDocument("*.doc", true) returns -1
- Position returns the header when the footer is in focus
- SaveDocument(string f,{FileTypeDoc!|FileTypeHTML!|FileTypePDF!}) returns -1 and FileExists event is triggered

7.8.5 Unsupported Features

- COM+ runtime
- Machine code generation
- TabletPC
- PBNI SDK for developing 64-bit PowerBuilder extensions
- DataWindow RichText style column
- DataWindow Web control for ActiveX
- Status bar
- Grid table
- ClearAll() function
- Clear(true) function
- Change Pointer does not work on RichTextEdit controls
- Mouse wheel does not scroll on a RichTextEdit page
- Application server support

NOTE: If you select Properties in the RichTextEdit Object Dialog popup menu, the application crashes if you select the Print Spec tabpage and click OK.

7.8.6 Behavior Differences

The RichText preview mode behaves differently; in 64-bit, it is more like a print preview

7.8.7 PowerBuilder Native Interface (PBNI)

You can only use 32-bit PowerBuilder extensions in the PowerBuilder Classic IDE. For runtime, package and distribute 64-bit extension libraries with your 64-bit applications. The

file names of your 64-bit extension should match the 32-bit file names, since the application references it by file name.

7.8.8 OrcaScript

To build 64-bit native applications with OrcaScript, use the new X64 option to build executable commands.

Syntax:

```
build executable <exeName> < iconName> <pbrName> <pbdflags> <machinecode>
 <newvstylecontrols> x64
```

7.9 New option for OrcaScript command

As of PowerBuilder 12.5, in addition to full, migrate and incremental, there is a new option: 3pass

This option performs a full rebuild of the application. You should use it instead of full when there are complex inherited relationships.

7.10 pbm_custom[xx] argument changed from uLong to LongPTR

The argument for pbm_custom01 through pbm_custom75 has been changed from data type uLong to LongPTR since PowerBuilder 12.6. After the scripts are migrated to version 12.6 or later, the user events will lose their mappings to pbm_custom[xx]. The developer will need to edit the object source code to map the user events to pbm_custom[xx] again and update the argument.

8 Upgrading from PowerBuilder 12.5 or earlier

8.1 Web Forms Target Removed

PowerBuilder 12.5.2 has removed the Web Forms target. If you are using Web Forms in PowerBuilder 12.5.1 or earlier, you should not install 12.5.2 GA. The EBFs for 12.5.2 will include fixes for Web Forms as well as issues for other features.

9 Upgrading from PowerBuilder 12.1 or earlier

9.1 RTF and Images in the DataWindow Object

PowerBuilder 12.5 introduces the RTF, Image, and XPS Database Blobs, as well as RichText and RichTextFile expression functions for Computed Fields.

The RTF, Image, and XPS Database Blob feature is a port of the existing feature from PowerBuilder .NET.

The RichText expression function takes as argument a string expression interpreted as RTF file and renders it as such. If the argument is not RTF, nothing renders.

The RichTextFile takes as argument a string expression interpretation of the name of the RTF file and renders it as such. If the argument is not an RTF file, nothing renders.

9.2 User-Drawn Controls in DataWindow Objects

The Paint (*expr*) expression functions allow you to draw objects in the DataWindow such as polygons, arrow tips, pie slices, and so on. The Paint expression function takes one string expression argument and returns the same string. This allows you to paint inside a DataWindow in a way that respects the position and z-order of other DataWindow objects. It should contain a function call to a drawing global function with rendering logic. If *expr* is a string expression and the value is not null, the Computed Field will render the evaluated string expression.

This feature also provides the following supporting functions:

- GetPaintDC()
- GetPaintRectX()
- GetPaintRectY()
- GetPaintRectWidth()
- GetPaintRectHeight()

9.3 Window Control transparent Value and transparency property

PowerBuilder 12.5 introduces two new features for window controls:

- transparent value for window control BackColor
- transparency property for window control

The following controls support the new value and property:

Table 9.1:

PictureButton	CheckBox	RadioButton	StaticHyperLink
GroupBox	SingleLineEdit	EditMask	MultiLineEdit

RichTextEdit	DropDownListBox	DownDownPictureListBox	ListView
TreeView	Tab	Graph	UserObject

9.4 Support for New ASE 15.5 Datatypes (IM)

PowerBuilder and InfoMaker supports the new datatypes in ASE 15.5: BIGTIME and BIGDATETIME.

- BIGTIME: Includes the hour, minute, second, and fraction of a second. The fraction is stored to six decimal places.
- BIGDATETIME: Includes the year, month, day, hour, minute, second, and fraction of a second. The fraction is stored to six decimal places.

9.5 Sharing Datasources with .NET

New support is added for sharing ADO.NET connections. PowerBuilder 12.5 provides the ability to share ADO.NET connections between PowerBuilder (Win32) and third-party .NET assemblies exposed in COM. A PowerBuilder native application provides the capability to import an ADO.NET connection from a third-party .NET assembly and to export an ADO.NET connection to a third-party assembly. This feature already exists in PowerBuilder.NET, but the PowerBuilder Classic IDE supports a native version of this feature using .NET COM Interop. Two methods have been added to the PowerBuilder Transaction NVO:

- bool SetAdoConnection(oleobject connectionProxy) Imports an external ADO.NET connection.
- oleobject GetAdoConnection() Exports an ADO.NET connection from a connected PB Transaction instance.

You can invoke these two method to get or set the proxy to share an ADO.NET DB Connection between a PB app and third-party .NET assembly.

9.6 AutoWidth Property (IM)

In the Grid presentation style report, the AutoWidth property allows you to choose the option to automatically compute the width of a column. The AutoWidth property takes one of these numeric values:

- 0 - No AutoWidth: This is the default value.
- 1 - AutoWidth is computed for visible rows (monotonic) and does not decrease when the widest column is reduced when scrolling.
- 2 - AutoWidth is computed for visible rows (non-monotonic)
- 3 - AutoWidth is computed for all retrieved rows.

You can set the AutoWidth property:

- In the painter - in the Properties view, select one of the values in the drop-down list for the AutoWidth property.
- In scripts - set the AutoWidth property to one of the numeric values.

9.7 Support for Tab Sequence, Enabled, and Show Focus Rectangle Properties

These properties are supported in the PowerBuilder Classic DataWindow. PowerBuilder .NET WPF DataWindow supports only Tab Sequence and Enabled. The Tab Sequence property is not expressionable, but Enabled and Show Focus Rectangle are.

- The default value of Tab Sequence is zero for all non-column controls, replicating the previous behavior. You can edit it as needed.
- The default value for the Enabled property is "yes."
- The default value of the Show Focus Rectangle property is "no."

Note: The Show Focus Rectangle property is supported only in PowerBuilder Classic.

The following DataWindow controls now have the Tab Sequence, Enabled, and Show Focus Rectangle properties:

- Button (Show Focus Rectangle does not apply)
- Computed Field
- Graph
- Picture
- TableBlob
- Text

The Column control does not have the Enabled property because it has existing mechanisms which achieve the same effect. PowerBuilder Classic supports these properties for OLE Objects and OLE Database Blobs.

10 Upgrading from PowerBuilder 12.0 or earlier

10.1 ASE 15.5 Support (IM)

In PowerBuilder and InfoMaker 12.1, you can access Adaptive Server Enterprise 15.5. However, no support for new ASE 15.5 features is added.

10.2 Sql Anywhere 12.0 support (IM)

In PowerBuilder and InfoMaker 12.1, you can access SQL Anywhere version 12.0. However, no support for new SQL Anywhere features is added.

10.3 HasMinHeight Property

PowerBuilder Classic 12.1 has a new height property and a new height expression function for the DataWindow object. This property is only supported in PowerBuilder Native. It is not supported in Windows Forms, Web Forms, or PowerBuilder .NET. HasMinHeight only applies to columns that have Autosize enabled. When both Autosize height and HasMinHeight are true, the height value will be the max value between the auto height value (the calculated value when the Autosize height is set to true) and the value specified in the Height property. For example:

```
dw_1.object.fname.Height.HasMinHeight = "no"
dw_1.object.compute_1.Height.HasMinHeight = "yes"
dw_1.object.t_1.Height.HasMinHeight = "no"
```

10.4 FontHeight Function

PowerBuilder Classic 12.1 has a new height expression function for the DataWindow object. This function is only supported in PowerBuilder Native. It is not supported in Windows Forms, Web Forms, or PowerBuilder .NET. The new DataWindow expression function FontHeight allows you to find the height of the font for a column or computed field. This function takes the column name as an argument. Use this function to set the minimum height to the size of the font.

For example:

```
dw_1.object.fname.Height = "0~tFontHeight(fname)"
```

10.5 Find Function

The DataWindow **Find** function now has a buffer argument. You can use the new argument to find data in the Delete! or Filter! buffer. The default value is Primary! The argument is optional.

11 Upgrading from PowerBuilder 11.5.1 or earlier

11.1 Deprecated functionality

The ability to build a COM or COM+ component is no longer available. However, you can still connect to COM and COM+ components from standard PowerBuilder client-server applications.

11.2 PowerBuilder Classic and PowerBuilder .NET

PowerBuilder 12.0 installs with two separate IDEs. The familiar PowerBuilder IDE is rebranded as PowerBuilder Classic. The new IDE is called PowerBuilder .NET.

The PowerBuilder Classic IDE retains the same basic functionality as in earlier PowerBuilder releases. The PowerBuilder .NET IDE hosts the Visual Studio isolated shell and is designed for compliance with the common language specifications for .NET.

PowerBuilder .NET includes one new target type (WPF Window Application) and two new project types (WPF Window and WCF Client Proxy). You can migrate PowerBuilder Classic client-server and Windows Forms targets to PowerBuilder .NET using the WPF Window Application target wizard.

The .NET Assembly target type is available in both PowerBuilder Classic and PowerBuilder .NET, but in PowerBuilder .NET, you can take advantage of language enhancements for fuller .NET compliance. You can also migrate .NET Assembly targets from PowerBuilder Classic to PowerBuilder .NET.

For information about PowerBuilder .NET targets and projects, see the PowerBuilder .NET Features Guide.

11.3 Platform support (IM)

PowerBuilder 12.0 adds support for the Windows 7 Professional 32-bit platform, and has also been tested on Windows XP (SP3), Windows XP Tablet PC (SP 3), Windows Server 2003 (SP 2), and Windows Vista (SP 2). PowerBuilder 12.0 maintains support for deployment to Windows Server 2008 (SP 2), but no longer supports deployment to Windows 2000.

11.4 New properties, functions and database parameters

The following new items are documented in the online Help for PowerBuilder 12.0:

- OriginalSize property for buttons and picture controls in DataWindow objects
- Clear (gridFlag) syntax for RichTextEdit controls
- ClearAll function for RichTextEdit controls
- CloseUserObject, OpenUserObject, and OpenUserObjectWithParm functions are supported for visual user objects
- PBAddCookie and PBGetCookies functions for Web Service proxy objects

- Eleven SoapPBCookie methods for getting and setting cookie properties
- Four SoapConnection methods for setting proxy server bypass conditions
- GenerateEqualIsNull database parameter for all database connections
- NCharBind database parameter for SQL Server (SNC) connections

11.5 Enhancements to the Runtime Packager

In PowerBuilder 12.0, the Runtime Packager is enhanced to allow generation of a Microsoft Merge Module (MSM file) instead of an MSI file.

If you select the PowerBuilder .NET Components option, the MSM or MSI file created by the Runtime Packager now includes the runtime files required for applications developed in the new PowerBuilder .NET IDE. However, the installation package you create with the PowerBuilder .NET Components option still includes the runtime files required for .NET applications and components that you develop in the PowerBuilder Classic IDE.

In the Runtime Packager, you can also select an option to include DLL files required for exporting graph or DataWindow data to a file with the Microsoft Excel format. The Sybase.PowerBuilder.DataWindow.Excel12.dll and PBDWExcel12Interop120.dll files are added to the MSM or MSI file when you select the MS Excel12 Support check box and create an installation package from the Runtime Packager. The Microsoft Excel format also requires .NET Framework 3.0 or later, but this must be installed separately on the runtime computer, since it is not included in the package generated from the Runtime Packager.

11.6 Enhancements for the ADO.NET Interface

The PowerBuilder 12 ADO interface has been extended to support ADO.NET providers. Now, you can connect at runtime to any data source that adheres to the ADO.NET 2.0 Common Provider model.

Table 11.1:

ADO .NET Provider	***Enhancements for the ADO.NET Interface***
ADO.NET for Oracle	**ODP.NET Driver Updates** Drivers for these ODP.NET versions are updated: • For Oracle 10g, the Oracle.DataAccess.dll driver was upgraded from Version 10.1.0.301 to 2.102.2.20 • For Oracle 11, the Oracle.DataAccess.dll Version 2.111.6.20 driver was added Both drivers are ADO.NET 2.0 compatible. The PowerBuilder ADO.NET interface no longer includes a driver for Oracle ODP.NET 9i. Users of that provider should migrate to Oracle ODP.NET 10g. **New Features of ODP.NET 2.0 for Oracle 10.2 and Earlier**

- **Client Identifier:** The client identifier is a predefined attribute for the Oracle application context namespace, USERENV. Like proxy authentication, the client identifier enables tracking user identities. However, unlike proxy authentication, the client identifier does not require separate sessions for the proxy user and end user. Also, the client identifier does not need to be a database user, and can be set to any string. Most important, the client identifier enables ODP.NET developers to use application context and Oracle Label Security, and to configure an Oracle Virtual Private Database (VPD) more easily. Configure the client identifier for Oracle ADO.NET data providers in the Driver Specific tab of the Database Profile Setup dialog.
- **Connection Pool Optimizations for RAC Databases:** An Oracle Data Provider for ADO.NET optimizes connection pooling for Real Application Cluster (RAC) databases by balancing work requests across Oracle RAC instances, based on load balancing advisory and service requirements. In addition, the ODP.NET connection pool can be enabled to proactively free resources associated with connections that have been severed when an Oracle RAC service, instance, or node goes down. Specify ODP.NET connection pool optimizations as arguments to the ProviderString DBParm parameter. You can enter driver-specific parameters at the bottom of the Connection tab of the Database Profile Setup dialog.
- **Large Object Retrieval:** You can retrieve entire columns of large object (LOB) data even if the select list does not contain a primary key, row id, or unique key. To use this enhancement, set the InitialLOBFetchSize property value to -1 for CLOB and BLOB objects.
- **LONG Retrieval:** You can retrieve entire columns of LONG and LONGRAW data even if the select list does not contain a primary key, row id, or unique key. To use this enhancement, set the InitialLONGFetchSize property value to -1.
- **XMLType:** The Oracle XMLType datatype is mapped to the PowerBuilder string type, with these limitations:
  - *XMLType* cannot be used in Where clauses within PowerBuilder Embedded SQL statements or in a DataWindow object.
  - *XMLType* columns cannot be selected directly by an Oracle cursor.
  - *XMLType* cannot be a parameter of a procedure or function, because PowerBuilder binds *XMLType* as a *string* type, but Oracle does not support that usage.
- **Client Access Through a Proxy:** With proxy authentication, the end user typically authenticates to a middle tier (such as a firewall), that in turn logs into the database on the user's behalf, 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 Connection tab of the Database Profile Setup dialog provides a Connect As dropdown control. To create a proxy connection, enter a different value that is not one of the predefined control items (Default, SYSOPER, and SYSDBA). • **Transparent Application Failover Notification:** Transparent Application Failover (TAF) notification enables an application connection to automatically reconnect to another database instance if the connection is severed. When a failover occurs, applications may wish to be notified. A new DBParm, SvrFailover, supports TAF notification. By default, SvrFailover is set to 0. If SvrFailover is set to 1 (true or yes), the transaction object invokes the DBNotification event when a failover occurs. **New Features for ODP.NET 2.0 for Oracle 11g** • **ODP.NET Configuration:** Developers can now configure ODP.NET using configuration files, including the .NET application configuration file, web.config, and machine.config. Settings in the machine.config file override the registry settings. The settings in the application configuration file or the web.config file overrides the values in the machine.config file. • **Additional Connection Pool Optimizations for RAC and Data Guard:** ODP.NET now cleans up the connection pool when the database down event is received from Real Application Clusters (RAC) or Oracle Data Guard. This is in addition to the events for which ODP.NET previously cleaned up the connection pool: node down, service member down, and service down. • **Windows-Authenticated User Connection Pooling:** You can now manage operating system-authenticated connections as part of ODP.NET connection pools, through Windows account management. • **Connection Pool Performance Counters:** ODP.NET publishes performance counters for connection pooling, which can be viewed using the Windows Performance Monitor. For PowerBuilder, the counters can be set in the Windows registry or in the application configuration file. The following ADO.NET 1.1 features are not supported: • **Oracle User-Defined Types:** PowerBuilder does not support UDT types. • **Bulk Copy Operations:** ADO.NET 1.1 enables applications to efficiently load large amounts of data from a table in one database to another table in the same or a different database. PowerBuilder does not support bulk copies; instead it uses pipelines for table copy operations.
ADO.NET for	**Drivers for these ADO.NET versions are updated:**

Adaptive Server Enterprise	• The ASE 12.5x ADO.NET driver, Sybase.Data.AseClient.dll, is updated from Version 1.1.411.0 to 1.1.670.0. The ASE 12.5x ADO.NET driver is ADO.NET 1.1 compatible, and does not support ADO.NET 2.0. • The ASE 15 ADO.NET driver is updated from Sybase.Data.AseClient.dll Version 1.15.50.0 to Sybase.AdoNet2.AseClient.dll 1.15.325.0. The ASE 15 ADO.NET driver is ADO.NET 2.0 compatible. **New Features for ASE 15** The ASE 15 ADO.NET driver supports these new ASE identity types: • *Bigint* identity • *Int* identity • *SmallInt* identity • *Tinyint* identity • unsigned *bigint* identity • unsigned *int* identity • unsigned *smallint* identity

ADO.NET for Microsoft SQL Server	**New Features for SQL Server 2005 and Earlier**
	• Large value types:
	  • *varchar(max)*
	  • *nvarchar(max)*
	  • *varbinary(max)*
	• *xml*, *varchar(max)* and *nvarchar(max)* are mapped to the PowerBuilder string type; varbinary(max) is mapped to the PowerBuilder blob type.
	• PowerBuilder supports SQL Server database mirroring, and a DBNotification event is fired when failover occurs. A new DBParm parameter, FailoverPartner, enables you to set the SQL Server failover partner server, as in the SQL Native Client (SNC) interface
	• Query notifications are not supported by the PowerBuilder ADO interface for SQL Server.
	New Features for SQL Server 2008
	The following SQL Server 2008 features are supported:
	• New datatypes:

	• *date* • *time* • *datetime2* • *varbinary(max)*(filestream) The SQL Server *date*, *time* and *datetime2* datatypes are mapped to PowerBuilder *date*, *time* and *datetime* types. *varbinary(max) (filestream)* is mapped to the PowerBuilder *blob* type. The maximum scale of *time* or *datetime2* is 6. • The new T-SQL commands support: • MERGE statement • Grouping sets • Row constructors • Table hints • The new T-SQL command works in the PowerBuilder ADO interface for SQL Server, as in the SNC interface. These SQL Server 2008 featues are *not* supported: • *datetimeoffset* datatype • Table-valued parameters
ADO.NET for SQL Anywhere	Connect to a SQL Anywhere database using an iAnywhere.Data.SQLAnywhere provider. PowerBuilder applications can perform all database related operations, such as exploring SQL Anywhere database objects like tables and procedures, and retrieving and updating data in the Database Painter.
ADO.NET for Informix	The Informix DATETIME HOUR TO SECOND type is treated as type TIME in PowerBuilder. Also, the TIME type column is displayed in the Database Painter as DATETIME, because the two variants of Informix DATETIME type are indistinguishable in the resultset schema. The IBM.Data.Informix driver does not support the BindSPInput dbparm.
ADO.NET for DB2	PowerBuilder supports DB2 using the System.Data.Odbc provider for both runtime and design time operations.

12 Upgrading from PowerBuilder 11.5 or earlier

12.1 Deployment support for the Windows 2000 platform (IM)

Due to customer request, PowerBuilder 11.5.1 includes deployment and runtime support for the Windows 2000 platform that was previously discontinued with the PowerBuilder 11.5 release. However, Windows 2000 is not supported as a development platform, and runtime support may be removed in future releases of PowerBuilder.

12.2 JDK 1.6 Support (IM)

PowerBuilder 11.5.1 applications and components include support for JDK1.6_02, that you can optionally install with the Sybase EAServer 6.2 setup program.

12.3 FIPS 140-2 certification (IM)

The PowerBuilder 11.5.1 development environment meets the encryption requirements of the Federal Information Processing Standard (FIPS) as outlined in Publication 140-2 of the United States government's National Institute of Standards and Technology. To meet these standards, PowerBuilder embeds the Certicom 5.x cryptographic modules. Support is certified for the PowerBuilder development environment only, not for runtime applications that are built with PowerBuilder. The FIPS 140-2 standard requires that passwords be encrypted. The PowerBuilder 11.5.1 user interface displays all passwords as strings of asterisks. Some files with passwords, such as database profiles, that you export from PowerBuilder 11.5.1, cannot be correctly imported into earlier versions of PowerBuilder because of the enhanced password protection. However you can still import files containing unencrypted passwords from earlier versions of PowerBuilder. When you save these files in or export them from PowerBuilder 11.5.1, the passwords are encrypted using a FIPS compatible method.

12.4 Support for Microsoft Office 2007 Excel formats (IM)

PowerBuilder 11.5.1 you can also save DataWindow and graph data in .XLSX and .XLSB (Excel12) format, with or without column headers. Because .NET Framework 3.0 or later must be installed on development and runtime computers to use this functionality, you cannot save data in these formats on Windows 2000 computers.

Other than the Windows 2000 platform restriction, the same level of support is provided as for Microsoft Office 2003. You can use the PowerScript SaveAs command with graphs or DataWindow controls to save data to Excel formats by setting the saveastype argument to XLSX! or XLSB! You can also export DataWindow data to Excel formats by selecting one of the Excel12 items from the drop-down list in the Save As dialog box that displays when you select the Save Rows As menu item in the DataWindow painter.

Excel support is dependent on the following strongly named assemblies:

- Sybase.PowerBuilder.DataWindow.Excel12.dll This assembly is installed in the GAC.
- PBDWExcel12Interop115.dll This assembly is installed in the Sybase/Shared/PowerBuilder directory.

These DLLs are deployed by the Runtime Packager when you select the MS Excel12 Support check box.

12.5 Support for SQL Anywhere 11.0 mirroring (IM)

PowerBuilder 11.5.1 allows you to take advantage of SQL Anywhere 11.0 database mirroring. Database mirroring is a configuration of either two or three database servers that cooperate to maintain copies of database and transaction log files. If the primary server becomes unavailable because of hardware or software failure, the mirror server negotiates with the SQL Anywhere arbiter server to take ownership of the database and assume the role of primary server.

12.6 Informix 11.5 Support (IM)

PowerBuilder 11.5.1 applications and components work correctly with the Informix 11.5 DBMS through the I10 interface, although new Informix 11.5 features are not supported in this release. You can use the 110 Informix v10.x database driver to connect to the Informix 11.5 DBMS, but this also requires that you upgrade the Informix client from Informix SDK 2.9 to Informix SDK 3.5.

12.7 Microsoft SQL Server 2008 support (IM)

PowerBuilder 11.5 included support for many of the new SQL Server 2008 features, but the released SQL Server product was not available for complete testing before the PowerBuilder release date. However, these features have been tested with PowerBuilder 11.5.1. They are described in the New Features Guide for PowerBuilder 11.5, and in the online Help.

13 Upgrading from PowerBuilder 11.2 or earlier

13.1 Directory changes for FDCC compliance (IM)

The Federal Desktop Core Configuration (FDCC) is a security standard mandated by the US Office of Management and Budget (OMB). 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, in PowerBuilder 11.5 and later, all writable files are installed, copied, or created in directories where standard users have write access.

13.1.1 FDCC constraints on certain PowerBuilder features (IM)

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.

13.1.2 Files shared by all users (IM)

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

For InfoMaker, writable files shared by all users are installed in the C:\Documents and Settings\All Users\Documents\Sybase\InfoMaker 11.5 directory on Windows XP and Windows 2003, and on Windows Vista and Windows 2008, in C:\Users\Public\Documents \Sybase\InfoMaker 11.5. This includes the:

- InfoMaker Windows Help and compiled HTML Help files

13.1.3 Files reserved for individual users (IM)

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 or InfoMaker. In this way, each PowerBuilder or InfoMaker user gets a private copy of these files.

Table 1 lists the files that are copied and updated in the directories of all users who run an instance of PowerBuilder or InfoMaker. The path variable in the table header (UserName) stands for the user name of a PowerBuilder or InfoMaker 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.

Table 13.1: New directory locations for some installed PowerBuilder and InfoMaker files

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
On Windows XP and 2003: Local Settings\Application Data\Sybase \InfoMaker 11.5 On Windows Vista and 2008: AppData \Local\Sybase\InfoMaker 11.5	• Initialization files (IM.INI, PBLAB115.INI, PBODB115.INI) • License files (IM115.LIC, im115_sysam.properties)
On Windows XP and 2003: My Documents\Sybase\InfoMaker 11.5\Tutorial On Windows Vista and 2008: Documents \Sybase\InfoMaker 11.5\Tutorial	• Files for the InfoMaker 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.

13.2 Deprecated Features

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

14 Upgrading from PowerBuilder 11.1 or earlier

14.1 AJAX update functionality for Web Forms applications

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

The AJAX enhancement for Web Forms applications does not require any changes in the PowerBuilder IDE, and you do not need to alter your PowerScript code to use the AJAX feature. However, PowerBuilder cannot deploy a Web Forms application unless AJAX is installed on the Web server. You can download and install the Microsoft ASP.NET AJAX Extensions version 1.0 on both the development and deployment computers from the ASP.NET website at http://www.asp.net/ajax/downloads/archive.

14.2 Telerik RadControls replace IE Web Controls in Web Forms applications

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

If you must use IE Web Controls, you need to download IE Web Controls from the Microsoft website and install the controls on the Web server. After you install the controls, you must set the PBWebControlSource selection to "IE", either on the Configuration tab of the Project painter before you deploy your application, or in the Web.config file after you deploy your application.

15 Upgrading from PowerBuilder 11.0 or earlier

15.1 Upgrading to Microsoft Vista (IM)

One of the most significant changes in Windows Vista is the introduction of User Account Control (UAC). UAC is the mechanism Vista uses to limit the default privileges users run with on Vista. Limiting these privileges is intended to make it less likely that a user's actions affect the configuration of the system or other users' state and settings.

15.1.1 Running PowerBuilder (IM)

PowerBuilder typically writes to multiple files in the Program Files directory, including pb.ini, PB110_sysam.properties, and code examples, tutorial, and demo database files. If an application is run without administrative privileges, Vista restricts it from writing to the HKEY_LOCAL_MACHINE subtree in the registry, the Program Files directory, and the Windows directory. Instead, changes are written to a writable area in the registry and to the user's local directory. This is referred to as virtualization.

For example, if you start PowerBuilder without administrative privileges, the pb.ini file is written to Program Files\Sybase\PowerBuilder 11.0\pb.ini in the C:\Users\<your_name>\AppData\VirtualStore directory.

To avoid the issues that would result from this behavior, on Vista, you must start PowerBuilder 11.1 with administrative privileges by right-clicking pb110.exe in the Start menu or Explorer and selecting Run as administrator from its pop-up menu.

15.1.2 Signed DLLs required for deployment (IM)

When you run an application on Vista, Vista warns you if the application uses any DLLs that have not been signed with an Authenticode certificate. If you want to deploy your application with Windows Vista Logo certification, all the DLLs that you distribute with it must be signed. PowerBuilder runtime DLLs, and all other PowerBuilder DLLs, are signed.

15.1.3 IIS 6 compatibility component requirement for Web Forms on IIS 7 (Vista)

If you deploy a Web Forms application to a remote server running IIS 7, or if you publish a smart client application to a local or remote server running IIS 7, the Vista Metabase Compatibility component of IIS 7 must be installed on the server. This component is not installed by default. It is not required to deploy a Web Forms application to a local server.

You can install it from the Programs and Features page in the Windows control panel. Select Turn Windows features on or off, then select Internet Information Services>Web Management Tools>IIS 6 Management Compatibility>IIS Metabase and IIS 6 configuration compatibility.

15.1.4 Permission requirements for Web Forms applications on IIS 7 (Vista)

When you deploy a new Web Forms target, a temp directory is created in the Inetpub\wwwroot\application_name directory, where application_name is the name of your application, and several subdirectories are created in the Inetpub\wwwroot

\application_name_root directory. Files are written to and deleted from these directories, therefore the IIS_IUSRS group must have full permissions on temp and application_name_root.

Before a PowerBuilder .NET Web Forms application connects to a SQL Anywhere database, you must either start the database manually or grant the IIS_IUSRS group on IIS 7 default permissions for the Sybase\Shared and Sybase\SQL Anywhere directories, making sure to replace permissions of all child objects in those directories. Full permissions are required for the database and database log files used by your application.

If you do not grant the appropriate user permissions for Sybase directories and your database configuration is set to start the database automatically, your application will fail to connect to the database. SQL Anywhere cannot access files unless the IIS_IUSRS user group has the right to access them.

15.1.5 Some calendar properties are not supported on Vista (IM)

The Vista operating system does not support several properties for the DatePicker, EditMask, and MonthCalendar controls and the drop-down calendar in a DataWindow column. The following properties are not supported on Vista:

- DatePicker properties: CalendarBackColor, CalendarFontName, CalendarFontWeight, CalendarItalic, CalendarTextColor, CalendarTextSize, CalendarTitleBackColor, CalendarTitleTextColor, CalendarTrailingTextColor, CalendarUnderLine
- EditMask properties: CalendarBackColor, CalendarTextColor, CalendarTitleBackColor, CalendarTitleTextColor, CalendarTrailingTextColor
- MonthCalendar properties: FaceName, MonthBackColor, TextColor, TextSize, TitleBackColor, TitleTextColor, TrailingTextColor, Underline
- Properties for column controls in DataWindow objects with a drop-down calendar EditMask style: DDCal_BackColor, DDCal_TextColor, DDCal_TitleBackColor, DDCal_TitleTextColor, DDCal_TrailingTextColor

In addition, the Vista operating system does not support the WeekNumbers property for the DatePicker control. When this property is true, the DatePicker control is not displayed correctly. The same limitation applies to the MonthCalendar control when WeekNumbers is true and Autosize is false.

15.1.6 JSP targets are not supported on Vista

On the Vista operating system, you can create a JSP target and a JSP page, but the component used to implement the HTML Editor's Page view and its built-in Script editor is not supported on the Vista operating system, therefore JSP targets are not supported on Vista.

16 Upgrading from PowerBuilder 10.5 or earlier

16.1 Upgrading EAServer targets

In PowerBuilder 11, the EAServer Component target wizard creates a specialized EAServer target instead of an Application target. After you migrate an existing EAServer target to PowerBuilder 11, you cannot start the remote debugger to debug the target unless you open the Project painter and select the Debug menu or toolbar item, or select Debug from the project's pop-up menu in the System Tree. To ensure that your target behaves correctly, you should use the EAServer Component target wizard to create a new EAServer target, select "Use an existing library and EAServer component project" in the wizard, and select your migrated library and component.

17 Upgrading from PowerBuilder 10 or earlier

17.1 Toolbar changes in PowerBuilder 10.5

In the Menu painter, you can now add a toolbar to a standalone main window as well as to an MDI frame. PowerBuilder adjusts the size of the main window to accommodate the toolbar. If your application currently uses a visual user object as a toolbar in a main window, the adjustments that PowerBuilder makes might affect the display of your toolbar and conflict with adjustments that your scripts make to display microhelp.

You can replace your toolbar user object with a toolbar designed in the Menu painter or continue to use your existing toolbar. To ensure that your existing toolbar displays correctly, set the window's ToolbarVisible property to false in a script or on the Toolbar page in the Properties view. To avoid conflicts, you should also move any microhelp position adjustment code into an event that runs after the Open event of the window.

17.2 Icon changes in PowerBuilder 10.5 (IM)

In PowerBuilder 10.5, many of the icons used in the PowerBuilder and InfoMaker user interfaces were changed. When you upgrade an application to PowerBuilder 10.5, any stock icons used in the application are updated automatically. For users who prefer to use the existing icons, a zip file that contains 24 icon files and more than 500 bitmap files used in previous versions of the products is available on the CodeXchange website.

17.3 Upgrading components to EAServer 6.0.1 or later

Intercomponent calls from a PowerBuilder component running in EAServer 6.0.1 require proxies for all called components. With earlier versions of EAServer, a PowerBuilder component is sometimes able to call another PowerBuilder component running in the same server without the use of a proxy, because the PowerBuilder VM creates a proxy for the component dynamically using method names that match the names of the component's methods.

In EAServer 6.0.1 and later, PowerBuilder components are wrapped as EJBs, providing an extra layer of security and preventing the PowerBuilder VM from generating a proxy with names that match the component's method names dynamically. Therefore, you must create a proxy object for all components you invoke with intercomponent calls. Without a proxy object, the TransactionServer object cannot obtain the correct method names of the component you are calling.

17.4 Creating an EJB client application for EAServer 6.x

Building EJB client applications for EJBs running in EAServer 6.x requires you to take some additional steps when you create the EJB client proxy and when you create the client.

To generate a proxy for an EJB deployed to EAServer 6.x:

1. Copy the packagename directory from the %DJC_HOME%\deploy\ejbjars\ directory on the server to the client computer, where packagename is the package that contains the EJB you want to use.

2. Add this directory to the Classpath on the Select EJB Component dialog box in the EJB Proxy Project painter.

3. Generate the proxy.

To create an EJB client application for an EJB deployed to EAServer 6.x:

1. Copy the eas-server-14.jar file (or eas-server-15.jar if you are using JDK 1.5.x) from the %DJC_HOME%\lib directory to the client computer and include its full path in the client's classpath.

2. Copy the stub files from %DJC_HOME%\genfiles\java\classes\ directory to the client computer and include this path in the client's classpath.

3. Copy the packagename directory from the %DJC_HOME%\deploy\ejbjars\ directory on the server to the client computer, where packagename is the package that contains the EJB you want to use and include this path in the client's classpath.

If you copied these files and directories to a directory on the client called EAServer6, and you want to use an EJB in the datamapping package, the client classpath setting might look like this:

Classpath=D:\EAServer6\lib\eas-server-14.jar;D:\EAServer6\genfiles\java\classes;D:\EAServer6\deploy\ejbjars\datamapping

17.5 PowerBuilder system types as variable names in proxies

In PowerBuilder 10.5 and later versions, system types cannot be used as variable names in Web service proxies. If a PowerBuilder system type is used as a variable name, the Web Service Proxy wizard renames the variable by applying the prefix ws_. If you are Upgrading Web service applications from PowerBuilder 10.2 or earlier and regenerating the Web service proxies in PowerBuilder 10.5 or later, your code may need to be modified to reflect the change in variable names.

PowerBuilder system types include not only the objects and controls listed on the System tab page in the PowerBuilder Browser, but also the enumerated types listed on the Enumerated page in the Browser, such as band, button, encoding, location, and weekday. For example, if you build a Web service from a PowerBuilder custom class user object, and one of its functions has a string argument named location, in the proxy generated for that Web service, the argument is changed to string ws_location.

17.6 OLE DB performance with Microsoft SQL Server

In PowerBuilder 10.5.2 and later, when you use the OLE DB database interface with a Microsoft SQL Server database and retrieve data into a DataWindow or use an embedded SQL cursor in a SELECT statement, server-side cursors are used to support multiple command execution. If this has a negative impact on performance, try increasing the size of the Block database parameter to 500 or more, or adding the following line to the [Microsoft SQL Server] section in the PBODB initialization file to turn off server-side cursors: ServerCursor = 'NO'

17.7 Change in behavior of OpenTab

A change was made in PowerBuilder 10.2.1 Build 9716, PowerBuilder 10.5.1 Build 6505, and PowerBuilder 11.0 Build 5021, to correct an anomalous behavior when the SelectedTab property was applied at runtime to a tab whose Visible property was set to false.

As a result of this change, there is a change in the behavior of the OpenTab and OpenTabWithParm functions. In previous releases, calling the OpenTab or OpenTabWithParm function to open a user object as a tab page displayed the tab page even if the user object's Visible property was set to false. In current releases, the user object's Visible property must be set to true for the tab page to display.

17.8 RichTextEdit control and RichText DataWindow changes (IM)

PowerBuilder 10.5 uses a new rich text editor to support the RichTextEdit control and the RichText DataWindow presentation style. The new editor brings a modern look and includes some new features, including the ability to name and use formatting styles. The new rich text editor supports a subset of the RTF specification version 1.6. Most of the properties and functions of rich text objects in previous versions of PowerBuilder continue to be supported by the new editor. When you import rich text objects from previous versions of PowerBuilder, any obsolete properties and functions are ignored.

There are some differences in behavior that may require you to make some changes in your applications. For example, when you upgrade applications created in older versions of PowerBuilder, the InputFieldsVisible property in RichTextEdit controls and in RichText DataWindow objects is automatically set to "false" in the upgraded applications. You must set this property to "true" to see data in the input fields. You must set this property and the InputFieldNamesVisible property to "true" to see text labels for the input fields in a rich text control.

For more information about changes, see the section on Rich text enhancements in the New Features in PowerBuilder 10.5 guide on the Sybase Product Manuals website.

17.9 Some PSR files must be regenerated (IM)

PSR files created in builds of PowerBuilder 10.0 or 10.0.1 prior to EBF build 6044 cannot be opened in later builds of PowerBuilder or InfoMaker. You must regenerate the PSR file in a later build.

18 Upgrading from PowerBuilder 9 or earlier

18.1 Unicode changes (IM)

PowerBuilder 10 is Unicode enabled. The source code in PowerBuilder 10 PBLs is encoded in UTF-16LE. UTF-16LE is a Unicode encoding scheme that serializes a UTF-16 code unit sequence as a byte sequence in little-endian format, in which multiple-byte numerical values are stored with the least significant byte first. PBLs developed in earlier versions of PowerBuilder contain source code in ANSI or DBCS format. When you upgrade applications to PowerBuilder 10, each PBL is first upgraded to the new version of PowerBuilder, as in previous releases. Then PowerBuilder converts the source code from ANSI or DBCS to Unicode, performs a full build, and saves the source code back to the same file.

As a result of this change, some new functions have been added and there are changes in the syntax of file-related functions and external function calls. For more information about these changes, see the Unicode support section in the New Features topic in online Help and the section on Unicode in *Application Techniques*.

18.2 Automatic changes made when you upgrade

When you upgrade an application from a previous release, the source code is converted to Unicode and the following changes to your source code are made automatically:

The clause ALIAS FOR *functionname* ;ansi is appended to external function declarations that return a string, char, or structure datatype or have a string, char, or structure value as an argument. This indicates that the arguments and/or return values should be treated as ANSI. If an ALIAS FOR clause is already present, only ;ansi is added. If ;ansi is not appended to the function name, strings are treated as Unicode.

The **FromAnsi**, **FromUnicode**, **ToAnsi**, and **ToUnicode** functions will be removed from a future version of PowerBuilder. The migration tool replaces them with the appropriate syntax of the **Blob** or **String** function.

No changes are made to the code if the PBL has already been upgraded to PowerBuilder 10.

In a DBCS environment, you can check the "Automatically convert DBCS string manipulation functions" check box on the Migrate Application dialog box to modify your code to comply with changes required for Unicode support. Do **not** check this box in an SBCS environment. If you check this box, the W suffix is removed from PowerScript string-manipulation functions such as **LenW** and **RightTrimW**, and an A suffix is added to the following PowerScript functions: **Fill** , **Left**, **Len**, **Mid**, **Pos**, **Replace**, and **Right**. The changes to string-manipulation functions described in the documentation also apply to DataWindow expression functions. However, the migration process does not modify DataWindow expression functions automatically.

18.3 ImportFile size limit

PowerBuilder 10.0 and later versions are Unicode enabled. If your application uses the ImportFile method to import very large text files (approximately 839,000 lines) into a DataWindow or DataStore, ImportFile returns the error code -15. Larger text files could be imported in ANSI versions of PowerBuilder.

NOTE: The release bulletins for earlier versions listed the size limit for large files as approximately 839,000 lines. The size limit depends on the number of columns, as well as the number of lines.

18.4 Upgrading Web and JSP targets

Web targets that use PowerDynamo cannot be migrated directly to PowerBuilder 10 and must be rewritten using an alternative model such as JavaServer Pages or Active Server Pages. For information about converting PowerDynamo websites created using PowerBuilder to JSP, see the Technical Document Converting 4GL Web Pages from PowerDynamo to JSPs.

Web and JSP targets that use the Web DataWindow and were created in earlier versions of PowerBuilder must be modified to use the HTMLGenerator100 component. Most other Web and JSP targets can be opened and deployed in PowerBuilder 10 without modification. See Section 13.5, "JSP object model changes" for an exception. As a precaution, you should make backup copies of the target directories before making modifications.

18.5 JSP object model changes

Global control variables in the JSP object model have been changed to local variables to make JSP pages thread safe. If you want to refer to another control in a server-side event, you must qualify the name of the control with the string "psPage". For example, in previous releases, the following code in a button's ServerAction event sets the context of a single-line edit control:

```
sle_1.value = "abc";
```

In PowerBuilder 10 (and PowerBuilder 9 build 7151 and later), use the following statement instead:

```
psPage.sle_1.value = "abc";
```

18.6 DBCS text in object properties is not converted correctly (IM)

DBCS applications can be migrated successfully on operating systems with a DBCS-compatible locale. However, on an operating system with an English locale, DBCS characters display as garbage characters when the font property of an object is set to a font that does not support DBCS characters. To work around this issue, change the font to Tahoma after migration. [CR 355908]

18.7 XML string encoding

In PowerBuilder 10, the XML parser cannot parse a string that uses an eight-bit character code such as windows-1253. For example, a string with the following declaration cannot be parsed:

```
string ls_xml ls_xml += '<?xml version="1.0" encoding="windows-1253"?>'
```

You must use a Unicode encoding value such as UTF16-LE.

18.8 Runtime errors in EAServer

In PowerBuilder 7, if a runtime exception was raised by a PowerBuilder component running in EAServer, the transaction was rolled back and the exception was thrown back to the client.

In PowerBuilder 8, the behavior was changed so that the transaction was committed before the exception was thrown back. In PowerBuilder 10, PowerBuilder 9.0.2, and EBF releases of PowerBuilder 8 and PowerBuilder 9.0.1 dated February 27, 2004 or later, the default behavior has been reverted to the behavior in PowerBuilder 7 and the transaction is rolled back.

In PowerBuilder 10, PowerBuilder 9.0.2, PowerBuilder 9.0.1 EBF Build 7066 and later releases, and PowerBuilder 8.0.4, you can control this behavior using the PBRollbackOnRTError environment variable. When this environment variable is set to 'y', 'yes', or 'true', the transaction is rolled back before the exception is thrown back to the client. [CR 319543]

18.9 Using masks with "as is" characters

You can define a mask that contains "as is" characters that always appear in the control or column. For example, you might define a numeric mask such as Rs0000.00 to represent Indian rupees in a currency column. In PowerBuilder 9.0.1 or later, you cannot enter a plus or minus sign to represent positive or negative numbers in a mask that contains "as is" characters. In previous releases, you could enter a plus or minus sign, but the resulting behavior was inconsistent in DataWindow columns.

The preferred method of creating a currency edit mask is to use the predefined **[currency(7)] - International** mask. You can change the number in parentheses, which is the number of characters in the mask including two decimal places. When you use this mask, PowerBuilder uses the currency symbol and format defined in the regional settings section of the Windows control panel. You can enter negative values in a column that uses a currency mask. [CR 309118]

18.10 Format of WMF files saved from DataWindows changed

In PowerBuilder 9.0, the format of WMF files created by saving a DataWindow object was changed to fix a crash issue. However, the fix removed the header information that allows the WMF file to be viewed in other applications. The format has been changed to restore the header information while preserving the fix. This change is in PowerBuilder 9.0 EBF builds 6096 and greater and in PowerBuilder 9.0.1, 9.0.2, and 10. [CR 292406]

18.11 MTS/COM+ components must be redeployed

Due to changes in the PowerBuilder VM in PowerBuilder 9.0.1, you must redeploy existing components to MTS or COM+ if you want to call them from PowerBuilder 9.0.1 and later clients. If you do not redeploy the components, calls to functions of the TransactionServer and ErrorLogging objects return incorrect values.

18.12 Change in Date function behavior

When you use the Date function with a string argument, PowerBuilder attempts to match the input string to a date format in the regional settings on the computer. In PowerBuilder 10 and later, if a complete match is not found, PowerBuilder attempts a partial match. For example, if you use Date('01-JAN-1900') and PowerBuilder finds the partial match (dd-MMM-yy), PowerBuilder parses the first two numbers of the year and gets 19. The 2-digit year is interpreted as a year between 1930 and 2029, and the date returned is 1/1/2019.

19 Upgrading from PowerBuilder 8 or earlier

19.1 Changed format of PSR files (IM)

The format of PSR files created in PowerBuilder was changed in order to improve data integrity for the **SaveAsAscii** function. As a result, PSR files created in newer builds of PowerBuilder cannot be opened in builds that predate this change. This change was made in PowerBuilder 8.0 build 7063 and PowerBuilder 7.0.3 build 10102.

19.2 Source code control changes

PowerBuilder 8 provided a direct connection to external SCC-compliant source control systems, and additional changes were made in PowerBuilder 9.0. Before upgrading a source-controlled project from PowerBuilder 8 or earlier to PowerBuilder 9 or 10, read the chapter on using source control in the PowerBuilder *User's Guide*.

19.3 ScrollToRow behavior change

The ScrollToRow method triggers the RowFocusChanging and RowFocusChanged events. In PowerBuilder 7, both events occurred after focus changed to the new row. This behavior was changed in PowerBuilder 9 so that RowFocusChanging could be coded to cancel the scroll. However, this change caused both events to be triggered before focus changed to the new row. In PowerBuilder 9.0.1 Build 7136 and later, the RowFocusChanging event is triggered before the scroll occurs, and the RowFocusChanged event is triggered after the scroll occurs. [CR 345104]

19.4 Web ActiveX deployment requirements (IM)

Microsoft stopped shipping the Microsoft Java VM as of the Windows XP SP 1a release and the Windows 2000 SP 4 release, and the Microsoft Java VM is not supported in PowerBuilder 9 or later. If you use the DataWindow Web control for ActiveX and your Web page uses a JDBC connection, the Web ActiveX has additional deployment requirements:

- The Sun JRE 1.2 or later must be installed on the client. Users can download the latest version of the JRE from the Sun Java website.
- The path to the file *jvm.dll* (...*JRE**bin**client* for JRE 1.4 or later and ...*JRE**bin**classic* for JRE 1.2 or 1.3) must be added to each user's system PATH environment variable.
- The following files must be in a directory in the client's system PATH environment variable: *pbjvm90.dll*, *pbvm90.dll*, and *libjcc.dll* for PowerBuilder 9, or *pbjvmXXX.dll* and *pbshrXXX.dll* for later versions.
- The *pbjdbc12XXX.jar* file, which contains class files required by the Web ActiveX, must be deployed to the client. You can deploy the JAR file by referencing it in the CODEBASE attribute of the Object element in your Web page.
- Java classes required by your database vendor's client layer must be available on the client. They can be added to a CAB file that is referenced in the CODEBASE attribute of

the Object element in your Web page. For example, if you are using Sybase jConnect to connect to a database, the *jconn2.jar* file should be included in the CAB file. If the client layer is provided in a JAR file, it can be referenced directly in the CODEBASE attribute.

20 Upgrading from PowerBuilder 7 or earlier

20.1 Adding targets to a workspace

Step 1 -- Make backup copies of the target and libraries.

Step 2 -- Use the Migration Assistant in the new PowerBuilder to check for obsolete syntax or the use of reserved words in your code.

Step 3 -- Optimize all PBLs in the PowerBuilder IDE 7 or earlier.

Step 4 -- Create a workspace in the new PowerBuilder.

Step 5 -- Add the target and libraries to the workspace.

If your application built in PowerBuilder 7 or earlier contains a target, you can add the target to your workspace. To add a target to the workspace, right-click the workspace and then select **Add Target**. After you select the target, the **Migrate Application** dialog box opens, allowing you to migrate the application to the new PowerBuilder.

If your application built in PowerBuilder 7 or earlier contains no target, you can create a new target in your workspace and then add the existing libraries to the target. To create a new target in your workspace, right-click the workspace and then select New > Target > Application, and then select the existing libraries. After that, the Migrate Application dialog box opens, allowing you to migrate the application to the new PowerBuilder. For information about using workspaces and targets, see Chapter 1 in the *User's Guide*.

20.2 Distributed PowerBuilder is not supported

PowerBuilder 7 was the last version of PowerBuilder that incorporated distributed PowerBuilder functionality. Sybase recommended the use of EAServer in place of distributed PowerBuilder for distributed and Web applications in PowerBuilder 7 and later.

The Transport object and its associated properties and methods are obsolete in PowerBuilder 8 and later and were removed from PowerBuilder 9. Additional properties and methods that were used for distributed PowerBuilder and are therefore obsolete include:

- **ConnectionBegin** and **ConnectionEnd** events on the Application object
- **GetServerInfo**, **RemoteStopConnection**, and **RemoteStopListening** functions on the Connection object
- **SetConnect** function for proxy objects
- **ConnectString** and **Trace** properties on the Connection object
- **ConnectionInfo** structure

The JavaBeans Proxy and Web.PB generators were also used with distributed PowerBuilder applications and were removed from the New dialog box.

20.3 Reserved words

New reserved words were added to the PowerScript language in PowerBuilder 8 to support exception handling. If you use any of the new reserved words (**TRY**, **CATCH**, **FINALLY**,

THROW, and **THROWS**) as identifiers in existing applications, you must change these identifiers, giving them nonconflicting names. You can run the Migration Assistant, available on the Tool tab page in the New dialog box, to locate incorrect use of the new reserved words.

20.4 SystemError event change

In PowerBuilder 7 or earlier, if an error occurs that is not caught by an Error or ExternalException event, the application's SystemError event is triggered immediately. If there is no code associated with the SystemError event, the application is terminated. Otherwise, after the SystemError event executes, control returns to the location in the code where the error occurred.

In PowerBuilder 8 and later, if an error occurs that is not caught by the exception handling mechanism or by an Error or ExternalException event, the script terminates and the call stack is unwound. If the error occurs as the result of a **Triggerevent** call in a script, the calling script terminates and the call stack is unwound. In most cases, the SystemError event is not triggered until the call stack becomes empty. If an event of a response window caused the error, the SystemError event is triggered as soon as the response window event completes.

Because of this change in behavior, code that follows the statement that caused the error is not executed after the SystemError event is fired, as it would have been in previous releases. This change has a major impact on applications that rely on the previous behavior of returning control to the script where the error occurred. Code that relies on this behavior must be modified in PowerBuilder 8 and later.

You can handle potential errors by wrapping code that might cause an error in a try-catch block to prevent the SystemError event from being triggered when an execution error occurs. It is still advisable to code the SystemError event to handle any uncaught exceptions. You should not allow the application to continue after the SystemError event is invoked. The SystemError event should clean up and halt the application.

20.5 IsValid function change

The **IsValid** function now returns **false** if passed an argument of type *Any* that cannot be converted to a PowerObject. In PowerBuilder 7 and earlier, passing an invalid object to **IsValid** caused a system error. You should also take note of the SystemError event change described in the previous section.

20.6 Format for color options changed (IM)

You can specify custom colors for each component of graphical table representations in the Database or SQL Select painter by selecting Design>Options>Object Colors. The colors you specify are saved in your *PB.INI* file in the [Database] section. The format in which these colors are stored changed in PowerBuilder 8 and later to support the increased number of Windows system colors and custom colors available for controls. For example, these are color definitions for lines representing keys in a PowerBuilder 7 *PB.INI*:

ForeignKeyLineColor=0 0 255 IndexKeyLineColor=255 0 0 PrimaryKeyLineColor=0 128 0

These are the corresponding entries in a PowerBuilder 8 or later *PB.INI*:

ForeignKeyLineColor=16711680 IndexKeyLineColor=255 PrimaryKeyLineColor=32768

If you plan to use your PowerBuilder 7 *PB.INI* file, or the [Database] section from it, with PowerBuilder 8 or later, you should first delete all the color settings in the [Database] section. If you do not, the colors used might make tables unreadable in PowerBuilder 8 or later. You can reset custom colors in the Object Colors page of the Database Preferences dialog box in PowerBuilder 8 or later.

20.7 Web DataWindow migration issues

You might have used workarounds to solve problems with Netscape rendering in releases prior to PowerBuilder 7.0.2 C3. Some of these workarounds might not work correctly in later releases because of improvements to Netscape rendering.

Specifically, if you used computed fields or text fields containing only spaces, the Web DataWindow generator now creates a table entry for these fields, making the table display twice as wide. If you see this behavior, delete these placeholder fields and use a more standard layout.

20.8 DataWindow method return values for empty DataObject property

In PowerBuilder 8.0.2 and later, the value returned when there is no DataWindow object assigned to a DataWindow control or DataStore has been standardized for the methods listed in Table 2. Some of these return values are different from the values returned in PowerBuilder 7 and earlier releases. SetFilter is included in the table, but its return value was changed from 1 to -1 in PowerBuilder 8.0.4, not PowerBuilder 8.0.2.

Table 20.1: Return values when no DataWindow object is assigned

Method	Return value
AcceptText	1
DeleteRow	-1
GetItemDate, GetItemDateTime, GetItemTime, GetItemDecimal, GetItemNumber, GetItemStatus	null
GetItemString	Empty String
InsertRow	-1
Retrieve	-1
SelectRow	1
SetFilter	-1
Update	1

20.9 ScrollNextRow and ScrollPriorRow behavior change

In PowerBuilder 8 and later, the DataWindow methods **ScrollNextRow** and **ScrollPriorRow** trigger these events in the order shown:

- RowFocusChanging

- RowFocusChanged
- ItemFocusChanged
- ScrollVertical

In PowerBuilder 7 and earlier, the ScrollVertical event was triggered before the other events. You should no longer use these methods in the ScrollVertical event. Doing so causes the same series of events to be triggered repeatedly until the last or first row in the DataWindow is reached. [CR 323263]

20.10 Changed behavior of OpenSheet functions

In PowerBuilder 8 and later, the **OpenSheet** and **OpenSheetWithParm** functions might throw a runtime error on failure instead of returning -1. For example, this occurs when the optional *windowtype* argument is invalid. To ensure that this error is trapped, wrap the call in a try-catch statement in addition to checking the return value:

```
integer li_ret
try li_ret = OpenSheet(w_child, "w_child_1", MDI_User, 2, Original!)
 if li_ret <> 1 then
 MessageBox("OpenSheet failed", "Check arguments")
catch (RuntimeError rt)
 MessageBox("OpenSheet failed", rt.GetMessage() )
// Handle error end try
```

21 Upgrading from PowerBuilder 6.5 or earlier

21.1 Nested reports in DataWindow objects renamed (IM)

In PowerBuilder 7 and later, every object in a DataWindow object must have a name. During migration, objects without names are assigned names based on the user-definable prefix settings, usually *dw_1*, *dw_2*, and so on. Since names are assigned sequentially, an object might be assigned a name already used by another object in the DataWindow object. This can cause unexpected behavior. For example, naming an unnamed report with a name already used for another DataWindow object, such as *dw_2*, could cause a **GetChild**/**ShareData** or **Retrieve** operation to find and use the wrong DataWindow object.

To work around this problem, select Design>Options in the DataWindow painter and modify the DataWindow object prefix on the Prefixes tab before Upgrading. After you migrate the DataWindow object, you can change the prefix back.

21.2 Icons for windows must be assigned

A window no longer inherits its icon from the application that contains the window. To use the application icon, you must explicitly assign it to the window after migration using the new enumerated value AppIcon!.

21.3 ListView and TreeView controls events changed

PowerBuilder 7 and later use Microsoft ListView and TreeView controls. As a result, you might see some changes in behavior that require you to remap some events. When you perform mouse actions, some events do not fire, and some fire in a different order from previous releases.

21.3.1 TreeView

In PowerBuilder 7 and later, the pbm_rbuttonup event does not fire, but is immediately followed by a pbm_tvnrclicked event (the stock RightClicked! event for a TreeView). Therefore, you can copy any code from pbm_rbuttonup to RightClicked! or have the RightClicked! event trigger whatever code exists in the pbm_rbuttonup. In PowerBuilder 6 *both* pbm_rbuttonup and pbm_tvnrclicked fire.

Additionally, in PowerBuilder 7 and later, using the right mouse button to select a previously unselected TreeView item causes the previous TreeView item to regain focus when the button is released. In PowerBuilder 6, using the right mouse button to select a TreeView item causes it to become permanently selected. To duplicate this behavior in PowerBuilder 7 and later, place the line of code **this.SelectItem(handle)** in the RightClicked! event of the TreeView before triggering (or otherwise executing) code from the pbm_rbuttonup event.

In PowerBuilder 7 and later, the pbm_tvnrdoubleclick event (the stock RightDoubleClicked! event) does not fire but is immediately preceded by a pbm_rbuttondblclk event. Therefore, you can copy any code from the RightDoubleClicked! event to the pbm_rbuttondblclk event or have the pbm_rbuttondblclk event trigger existing code in the RightDoubleClicked! event. Both events fire in PowerBuilder 6.

21.3.2 ListView

In PowerBuilder 7 and later, the pbm_rbuttonup event *does not* fire if you use right-click on a specific ListView item, but it *does* fire if you right-click in the white area of the ListView where there are no items. A new event, pbm_contextmenu, always fires when the right mouse button is released. Table 3 shows when events are fired in PowerBuilder 7 and later.

Table 21.1: Table 3: Events fired in ListView

Location	Action	Events fired
On an item in a ListView	Press right mouse button	pbm_rbuttondown
Release right mouse button	pbm_lvnrclicked (the stock RightClicked! event) pbm_contextmenu	
On an empty area of the ListView	Press right mouse button	pbm_rbuttondown pbm_lvnrclicked (the stock RightClicked!event) pbm_contextmenu
Release right mouse button	pbm_rbuttonup pbm_contextmenu	

You should place code that should be executed when an item is actually selected by the right mouse button in the pbm_contextmenu event. This is how PFC ListView objects work in PowerBuilder 7 and later. The code that you want executed when the right mouse button is released on the white area of the ListView should remain in the pbm_rbuttonup event. Because pbm_contextmenu is called twice when you right-click in the white area, you should put code in the RightClicked event to retain the index of the item that was selected. If no item is selected, then the index value will be zero and you should use that as a test in the pbm_contextmenu code to decide whether that code should be executed.

The following example assumes that you have declared a private instance variable of a TreeView standard class user object called *ii_item*. This statement is in the Clicked! event script:

```
ii_item = index
```

The pbm_rbuttonup event script should contain code to be executed when the right mouse button is released after being pressed in the ListView but **not** on an item in that ListView.

The pbm_contextmenu event script should contain code like the following:

```
IF ii_item > 0 THEN
// code to be executed when the right mouse
// button is released after being pressed on an
// item in the ListView
END IF
```

The **pfc_u_lv** and **pfc_uv_lvs** objects have been modified to use pbm_contextmenu instead of pbm_rbuttonup.

NOTE: The PowerBuilder and InfoMaker Release Bulletins, New Features Guides were used as sources for this document.