What's New

Appeon PowerBuilder® 2019 R3

Contents

1 New Features in PowerBuilder 2019 R3

About this chapter

This chapter introduces the new features in PowerBuilder 2019 R3.

1.1 Decoupling Runtime and IDE

1.1.1 Why decoupling Runtime and IDE

The PowerBuilder runtime files are now separated from the PowerBuilder IDE, as an independent component "PowerBuilder Runtime".

The decoupling of Runtime and IDE makes it possible to install and upgrade IDE and Runtime separately, and one IDE and native C/S application can work with multiple runtime versions without needing to upgrade IDE or re-compile the application. For example, in the previous versions, when there was a new release, especially a release with new features or bug fixes on the runtime, if the developer wants to test it, s/he will have to configure a new environment or back up the existing environment before installing the new release, to avoid affecting the existing application. And now that Runtime is separated as an independent component, multiple runtime versions can be installed on the same machine, developers can switch to the new runtime version with only one click in the IDE, and if any issue is found, s/he can switch back to the original runtime in seconds; or if testing goes successfully, s/he can distribute the new runtime files with the application executable and configure the application executable to use the new runtime immediately without re-compiling the application executable.

1.1.2 What changes made for the decoupling

The following changes have been made to decouple Runtime and IDE:

- The 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, pbdom190.pbd is renamed as pbdom.pbd.

  If your application uses pbwsclient.pbd, pbdom.pbd, pbsoapclient.pbd, pbejbclient.pbd, make sure to update the file name and path accordingly.

- All runtime DLLs that used to be installed to the "Shared" folder are now separately installed to the "Runtime [version]" folder and the "IDE" folder instead, in order to physically separate Runtime files from IDE files.

  The "Runtime [version]" folder contains the libraries required at runtime. Example location: C:\Program Files (x86)\Appeon\Common\PowerBuilder\Runtime 19.2.0.2556.

  The "IDE" folder contains the support files required by IDE. Example location: C:\Program Files (x86)\Appeon\PowerBuilder 19.0\IDE.

  If any external program has called the runtime file, you may need to change the code accordingly in order for the program to find the dependent DLLs successfully (view code examples).

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

  The 32-bit apps in 32-bit OS search for this registry: HKEY_LOCAL_MACHINE \SOFTWARE\Sybase\PowerBuilder Runtime

  The 32-bit apps in 64-bit OS search for this registry: HKEY_LOCAL_MACHINE \SOFTWARE\Wow6432Node\Sybase\PowerBuilder Runtime

  The 64-bit apps in 64-bit OS search for this registry: HKEY_LOCAL_MACHINE \SOFTWARE\Sybase\PowerBuilder Runtime

- The PowerBuilder IDE (and the application executable) can select which version of runtime files will be used, so it enables developers to easily maintain multiple projects. The IDE can compile for multiple runtime versions (both minor and major) through the System Options. See Section 9.2.4.3, “Selecting a version of PowerBuilder Runtime” in *Application Techniques* for detailed instructions.

1.1.3 Installing and switching runtime

"PowerBuilder Runtime" is always installed to %AppeonInstallPath%\Common \PowerBuilder\Runtime [version]. You can install multiple versions of "PowerBuilder Runtime" and select a compatible one in the PowerBuilder IDE. By default, the latest compatible PowerBuilder Runtime will be used by the PowerBuilder IDE. You can easily switch the versions of runtime files for the PowerBuilder IDE and the application executable in seconds.

To install PowerBuilder Runtime, see Section 9.2.4.2, “Installing PowerBuilder Runtime” in *Application Techniques* for instructions.

To select which PowerBuilder Runtime will be loaded by the PowerBuilder IDE and which one will be loaded by the native C/S application, see Section 9.2.4.3, “Selecting a version of PowerBuilder Runtime” in *Application Techniques* for instructions.

The PowerBuilder IDE can work with PowerBuilder Runtime at the same or earlier versions, for example:

Table 1.1:

IDE (including MRs)	Supported Runtime (including MRs)	Unsupported Runtime
2019 R3	2019 R3	2021 or later
2021	2019 R3, 2021	2022 or later
2022	2019 R3, 2021, 2022	2022 R2 or later
2022 R2	2019 R3, 2021, 2022, 2022 R2	2022 R3 or later
2022 R3	2019 R3, 2021, 2022, 2022 R2, 2022 R3	2023 or later

1.1.4 Related features

The following changes have been made in accordance to the decoupling of runtime and IDE.

- A new property RuntimePath is added to the Environment object to get the runtime path and version used by the current application executable.
- A system function GetInstalledRuntimes is added to get the version number of runtimes that are installed on the current computer.
- A system option "Show prompt for full-building a target if it is opened after PowerBuilder Runtime change" is added -- Every time after you changed the version of PowerBuilder Runtime for the current IDE and restarted IDE, you will be prompted to full build the application. In most cases, it is recommended to full build your application every time after the runtime version is changed, unless your application is large and takes a long time to full build, then you can clear this check-box and manually full build the application only when there is a major version change for PowerBuilder Runtime. And once you performed a full build in the IDE, the new runtime version will be recorded in "appruntimeversion" which is an internal property used by the IDE only. (Note creating an exe from PBC, OrcaScript or IDE will not update the version number in the "appruntimeversion" property.)
- When using OrcaScript commands, you can use the *runtime_version* argument to specify the version of PowerBuilder Runtime that will be used to compile the application executable, for example, OrcaScr190 /D runtime_version="19.2.0.2558" C:\test\test.bat. For more, refer to Section 8.3.2, “OrcaScript Commands” in *Users Guide*.
- When using PowerBuilder Compiler (PBC), you can use the */rt* argument (for example, /rt 19.2.0.2558) to specify the version of PowerBuilder Runtime that will be used to compile the application executable. And PowerBuilder Compiler must be installed after PowerBuilder Runtime is installed. For more, refer to Section 8.2, “Appendix B. PowerBuilder Compiler” in *Users Guide*.
- The MSI/MSM file generated by the PowerBuilder Runtime Packager 2019 R3 tool has been enhanced, so that runtime files of different builds at the same major version (starting from 2019 R3 GA) can be installed and coexisting on the same computer, for example, 2019 R3 and 2019 R2 can coexist, multiple 2019 R3 MRs can coexist. And the MSI file no longer sets the runtime file path in the system PATH environment variable; therefore, the user will need to decide which build of runtime files will be loaded by the application executable file and place the application executable and the runtime files in the same folder.

1.1.5 Code examples

Example 1:

For example, to load the runtime DLL file PBORC190.dll from where PowerBuilder IDE is installed, you write code differently between 2019 R3 and 2019 R2.

Also note that the file name PBORC190.dll in 2019 R2 has been changed to PBORC.dll in 2019 R3.

Original code in 2019 R2:

```
hORCALibrary = AfxLoadLibrary("PBORC190.dll"))
```

Modified code in 2019 R3:

This code example hard-codes the path of IDE and runtime, if you want to read the directory from the registry, use the next example.

```
//First, defines the AddEnvironmentPath method
BOOL AddEnvironmentPath(LPCTSTR lpAddPath)
{
   TCHAR szSysPath[1024 * 8] = { 0 };
   TCHAR szNewPath[1024 * 8] = { 0 };

   if(0 == GetEnvironmentVariable(_T("path"), szSysPath, 1024 * 8))
      return FALSE;

   _tcscpy_s(szNewPath, lpAddPath);
   _tcscat_s(szNewPath, _T(";"));
   _tcscat_s(szNewPath, szSysPath);

   if(0 == SetEnvironmentVariable(_T("path"), szNewPath))
      return FALSE;

   return TRUE;
}

//Adds the path of runtime file before loading the runtime DLL
//The path needs to be updated when PB is upgraded
AddEnvironmentPath(_T("C:\\Program Files (x86)\\Appeon\\Common\\PowerBuilder\
\Runtime 19.2.0.2566"));
AddEnvironmentPath(_T("C:\\Program Files (x86)\\Appeon\\PowerBuilder 19.0\\IDE\
\"));
(hORCALibrary = AfxLoadLibrary("PBORC.dll")
```

Or you can write code like this in 2019 R3 which reads the path of IDE and runtime from the registry instead of the hard-coded path:

```
//First, defines the AddEnvironmentPath method
//AddEnvironmentPath(_T("19.2.0.1080"), _T("19"));
//AddEnvironmentPath(_T("19.2.0.1080"), NULL);
BOOL AddEnvironmentPath(LPCTSTR lpRuntimeVersion, LPCTSTR lpIDEShortVersion)
{
    LONG lRet = 0;
    HKEY hkReg = NULL;
    TCHAR   szIDEPath[MAX_PATH] = { 0 };
    TCHAR   szRTPath[MAX_PATH] = { 0 };
    TCHAR   szIDEName[MAX_PATH] = { 0 };
    TCHAR   szTmpRTPath[MAX_PATH * 2] = { 0 };
    TCHAR   szTmpIDEPath[MAX_PATH * 2] = { 0 };
    TCHAR   szSysPath[1024 * 8] = { 0 };
    TCHAR   szNewPath[1024 * 8] = { 0 };
    TCHAR   szRegPath[MAX_PATH] = { 0 };
    DWORD   cbLen = MAX_PATH * sizeof(TCHAR);
    DWORD   dwType = REG_SZ;

    //Reads runtime path from registry
    if (NULL == lpRuntimeVersion)
        return FALSE;

    wsprintf(szRegPath, _T("%s\\%s"), _T("Software\\Sybase\\PowerBuilder Runtime"),
 lpRuntimeVersion);
    lRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, szRegPath, 0, KEY_QUERY_VALUE, &hkReg);
    if ((ERROR_SUCCESS != lRet) || (NULL == hkReg))
        return FALSE;

    lRet = RegQueryValueEx(hkReg, _T("Location"), 0, &dwType, (LPBYTE)szRTPath,
 &cbLen);
    RegCloseKey(hkReg);
    if (_tcslen(szRTPath) <= 0)
        return FALSE;

    wsprintf(szTmpRTPath, _T("%s;%s\\X64;"), szRTPath, szRTPath);

    //Reads IDE path from registry
    if (lpIDEShortVersion && _tcslen(lpIDEShortVersion) > 0)
    {
        hkReg = 0;
        wsprintf(szRegPath, _T("%s\\%s.0"), _T("Software\\Sybase\\PowerBuilder"),
 lpIDEShortVersion);
        lRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, szRegPath, 0, KEY_QUERY_VALUE,
 &hkReg);
        if ((ERROR_SUCCESS != lRet) || (NULL == hkReg))
            return FALSE;

        cbLen = MAX_PATH * sizeof(TCHAR);
        lRet = RegQueryValueEx(hkReg, _T("Location"), 0, &dwType,
 (LPBYTE)szIDEPath, &cbLen);
        cbLen = MAX_PATH * sizeof(TCHAR);
        lRet = RegQueryValueEx(hkReg, _T("IPS Name"), 0, &dwType,
 (LPBYTE)szIDEName, &cbLen);
        RegCloseKey(hkReg);

        if (_tcslen(szIDEPath) <= 0 || _tcslen(szIDEName) <= 0)
            return FALSE;

        wsprintf(szTmpIDEPath, _T("%s\\%s\\IDE;%s\\%s\\IDE\\X64;"), szIDEPath,
 szIDEName, szIDEPath, szIDEName);
    }

    if (0 == GetEnvironmentVariable(_T("path"), szSysPath, 1024 * 8))
        return FALSE;

    _tcscpy_s(szNewPath, szTmpRTPath);
    _tcscat_s(szNewPath, szTmpIDEPath);
    _tcscat_s(szNewPath, szSysPath);

    if (0 == SetEnvironmentVariable(_T("path"), szNewPath))
        return FALSE;

    return TRUE;
}
```

In summary, you need to write code in two steps in version 2019 R3:

Step 1: Define a method that can add a path to the PATH environment variable.

Step 2: Call the method to add the PowerBuilder Runtime and PowerBuilder IDE directories to the PATH environment variable and then load the DLL file.

Note that if the path of IDE or runtime is hard-coded in the scripts, make sure to update the path accordingly if PowerBuilder Runtime and IDE are upgraded to a new version.

Example 2:

If you have source code like below which determines the PowerBuilder versions according to the class name, you will need to change the code accordingly.

Original code in 2019 R2:

```
Environment le_env
```

```
GetEnvironment(le_env)

//Determines position of window within microhelp bar
ii_width = st_time.x + st_time.width + 25
this.width = ii_width

//Sets object class name
choose case le_env.PBMajorRevision
       case 10, 11, 12
              choose case le_env.PBMinorRevision
                     case 5
                           is_classname = "FNHELP" + &
                                  String(le_env.PBMajorRevision) + "5"
                     case 6
                           is_classname = "FNHELP" + &
                                  String(le_env.PBMajorRevision) + "6"
                     case else
                           is_classname = "FNHELP" + String(le_env.PBMajorRevision
 * 10)
              end choose
       case else
              is_classname = "FNHELP" + String(le_env.PBMajorRevision * 10)
end choose

//Sets parenthood
this.wf_setparent()
```

Modified code in 2019 R3:

```
choose case le_env.PBMajorRevision
         case 10, 11, 12
                   choose case le_env.PBMinorRevision
                            case 5
                                     is_classname = "FNHELP" + &
                                               String(le_env.PBMajorRevision) + "5"
                            case 6
                                     is_classname = "FNHELP" + &
                                               String(le_env.PBMajorRevision) + "6"
                            case else
                                     is_classname = "FNHELP" +
 String(le_env.PBMajorRevision * 10)
                   end choose
         case 17
                   is_classname = "FNHELP" + String(le_env.PBMajorRevision * 10)
         case 19
                   if le_env.PBMinorRevision > 1 then
                            // pb2019 R3 or later:
                            is_classname = "FNHELP"
                   else //pb2019 R2 or earlier:
                            is_classname = "FNHELP" + String(le_env.PBMajorRevision
 * 10)
                   end if
         case IS > 19
                            is_classname = "FNHELP"
end choose
```

```
Function long GetClassName ( &
       longptr hWnd, &
       Ref string lpClassName, &
       long nMaxCount &
       ) Library "user32.dll" Alias For "GetClassNameW"
```

```
public subroutine wf_setparent ()
String ls_name
LongPtr ll_hWnd
```

```
Integer li_rc

//Gets the microhelp handle
ll_hWnd = GetWindow(Handle(gw_frame), 5)
DO UNTIL ll_hWnd = 0
       ls_name = Space(25)
       li_rc = GetClassName(ll_hWnd, ls_name, Len(ls_name))
       If ls_name = is_classname Then
              ll_hWnd = SetParent(Handle(this), ll_hWnd)
              ll_hWnd = 0
       Else
              ll_hWnd = GetWindow(ll_hWnd, 2)
       End If
LOOP
```

1.2 PowerClient deployment

PowerBuilder 2019 R3 introduces a new project type: PowerClient. PowerClient changes the way of deploying and updating PowerBuilder applications. Regardless your application is client/server or cloud-native architecture, the PowerBuilder client is deployed from a web server over HTTP/HTTPS, and it can update itself automatically. This eliminates the headaches and costs of creating installation programs, deploying apps to users, and keeping apps updated.

Key Features:

- Seamless installation, even without admin rights
- Self-updating with flexible update strategies
- Packages all necessary files (PBVM, OCXs, DLLs, etc.)
- Automatic file encryption and integrity verification
- Compatible with any Web server on any operating system
- Deploy to the Web server over FTP or create .ZIP package

Cloud App Launcher:

The Cloud App Launcher is a launcher program that must be first installed to the user's machine. It facilitates the initial installation and subsequent updates of your PowerBuilder applications over HTTP/HTTPS.

There are two launcher types. We recommend you test carefully to see which best meets your technical requirements:

- Launcher without background service: This launcher program does NOT use a background service. As such, it should be easier to install and use and does not require administration rights. However, it has certain dependency on the browser, which may result in different installation experience depending on the browser used and its configuration.
- Launcher with background service: The launcher program uses a background service. If there are multiple users on a client machine, the launcher requires administrator rights to install and will work together with the background service. This launcher type does NOT have dependency on the browser.

Compilation Differences:

PowerClient only supports compiling as p-code files. The compilation is different from the traditional p-code approach. All PBD files are broken down very granularly into each individual object/definition file. For example, each SRW, SRD, SRU, etc. file would have its individual corresponding p-code file (that have new file extensions, such as .dwo, .apl, .fun, .win, .udo) instead of a monolithic PBD files. Also, by default the p-code files are encrypted so they cannot be readily decompiled.

For more, refer to this whitepaper: Automating On-Premise Deployment of PowerBuilder Apps.

With PowerClient, you can

- Define a PowerClient project
- Build and deploy the PowerClient project
- Install and run the PowerClient project
- Package the PowerClient project
- Uninstall the PowerClient project

For detailed instructions, refer to Section 7.3.7, “Creating a PowerClient project” in *Users Guide*.

For a walkthrough of creating a PowerClient project, refer to Section 7.3.7.10, “Tutorial: deploying your first PowerClient project” in *Users Guide*.

Note

You must have a Professional or CloudPro license to use the PowerClient-related features.

Note

You must run PowerBuilder IDE as administrator in order to use the PowerClient-related features.

1.2.1 Related features

Two system functions IsPBApp and IsPowerClientApp are added to check if the current application is a native C/S application compiled using p-code/machine code or an application deployed using PowerClient.

1.3 Source control enhancements

Source control managements in PowerBuilder have been improved in these aspects:

- (Git and SVN) The "Refresh all PBLs with the source code after download" option is added to the Connect to Workspace dialog. You can select it to refresh all PBLs after downloading the source code from workspace if you are unsure whether the PBLs are updated with the latest source code.

- (Git only) You can create and switch branches in the PowerBuilder IDE, just like how you create and switch branches using TortoiseGit. For details, see Section 1.3.3.11, “Use branches” in *Users Guide*.

- (Git only) The branch information is shown in the PowerBuilder IDE, for example, the name of the current branch is displayed in the Commit menu, in the Commit dialog title, and in the output window.

- (Git and SVN) You can move the project along with its source control bindings to a new location without needing to bind the project with the source control server again. Before 2019 R3, when the project is moved to a new location or when PowerBuilder IDE is re-installed, the existing source control bindings become invalid, and you will have to connect with the source control server and download the project to re-establish the binding. In 2019 R3 (and later versions), after you move the project (including the .svn or .git and ws_objects sub-folders), you can continue with the source control management immediately.

  Besides that, if a project is under source control using a third-party tool such as TortoiseSVN or TortoiseGit, you can move it (including the .svn or .git sub-folder) and then select the source control type (SVN or Git) in the Properties of Workspace | Source Control tab to enable PowerBuilder IDE to add and continue with the source control management. For more, see Section 1.3.3.12, “View/Edit the connection settings” in *Users Guide*.

- (Git and SVN) You can specify the encoding format for the source code files in "ws_objects", when you add the workspace to the source control server. You can select from: ANSI/DBCS, HEXASCII, and UTF8.

- (Git and SVN) You can now use the Edit Conflicts option to edit conflicts through a third-party tool. You must first specify the third-party tool: in the Properties of Workspace dialog box, select the Source Control tab, and then click the Advanced button. In the Source Control Advanced Settings dialog box, select Show Log/Edit Conflicts in the left panel, and then specify the executable program of TortoiseSVN or TortoiseGit. For details, see Section 1.3.3.9, “Tools for Show Log\Edit Conflicts” in *Users Guide*.

  You should not commit objects if they are in conflicts. And after you edit the conflicts, you should refresh the objects to import the changes. It is also recommended that you regenerate or full build your project if there is any error after you resolve the conflicts.

- (Git only) PowerBuilder IDE will add the autocrlf option and set it to true on Windows if no other Git tool has installed and configured such an option. However, if you have manually changed the setting of autocrlf (from true to false or input), and then downloaded objects from the server, these objects will have LF line endings instead of CRLF, which will cause compilation errors in the PowerBuilder IDE. If such compilation error occurs, you should set autocrlf to true, download files from the server again, and then compile again. For more, Section 1.3.3.4, “Get objects from Git” in *Users Guide*.

- (Git and SVN) Project directory can contain multi-byte characters.

- (Git and SVN) The error messages occurred during the source control management in PowerBuilder IDE will be listed in the Errors tab of the Output window. Previously, the errors are only listed in the Default tab of the Output window.

1.4 License enhancements

PowerBuilder licensing mechanism has been improved in the following aspects:

- (Online and offline license) Starting from PowerBuilder 2019 R3, one user account can log into more than one version on the same machine simultaneously, just like logging into the multiple instances of the same version.

  Previously, if your user account has been used to log into one version (say 2019 R2), then this same account cannot be used to log into another version (say 2017 R3) on the same machine, unless this account is logged out from 2019 R2 first. While in 2019 R3 and later versions, the same account can be used to log into more than one version on the same machine simultaneously. For example, if you have installed and logged into 2019 R3 successfully, and then if you installed 2021, you can automatically log into 2021 using the same user account of 2019 R3, without needing to log out from 2019 R3 first.

- (Online license only) You can now start and log into PowerBuilder IDE by specifying your user account and password in the command line parameters. The command line mode can only work with the online license.

  You can directly specify your user account through the command line parameter. For example,

  ```
  pb190.exe /AC test@appeon.com /PW xxxxxxx /RC N /ALS N /SOE Y
  ```

  /AC -- User Account.

  /PW -- User Password.

  /RC -- Remember Credentials (Y or N), default is Y. /RC is always Y when /ALS is Y. Therefore, the /RC value will be ignored, if /ALS is set to Y. This parameter takes effect only when both /AC and /PW are set.

  /ALS -- Auto Login at Startup (Y or N), default is Y. This parameter takes effect only when both /AC and /PW are set.

  /SOE -- Sign Out on Exit (Y or N), default is Y. This parameter takes effect only when both /AC and /PW are set.

  You can also specify a file (that contains your user credentials) through the command line parameter. For example,

  ```
  pb190.exe /LIF c:\test.ini
  ```

  /LIF -- The full path to the license login initialization file which contains your encrypted password and other login settings. The file is created by an independent tool (%AppeonInstallPath%\PowerBuilder [version]\Tools\LoginIniFileCreator.exe). You can specify the user account, the password, the login settings, and the file path in this tool. The password will be encrypted in the generated file to protect your login credentials. (If you want to copy the tool to other places, make sure to copy the executable file as well as the msvcp100.dll and msvcr100.dll files under the same folder).

Note

If a user account has already successfully logged in (and not logged out since then), then only its user name and password will be authenticated when specified in the command line; the other settings (such as Auto Login at Startup, Sign Out on Exit, Remember Credentials etc.) specified in the command line will be ignored.

- Offline license are not allowed to be activated in the Windows Server operating system any more.

For more information, refer to the Appeon License User Guide.

1.5 Demo app changes

PowerBuilder 2019 R3 has removed the following legacy demo apps: Benchmark, DWGradientTransparency, DWRichTextEditStyle, MDIDockingDemo, Mobilink, and TreeView DataWindow and DatePicker; and provides only the following three demo apps. You can access them from Windows Start | Appeon PowerBuilder 2019 R3.

- **Example App** -- This demo app is the same as before.
- **Example Sales App** -- This demo app is newly added in 2019 R3. It demonstrates
  - how to use the RESTClient object to exchange data with the RESTful Web APIs
  - how to modernize the UI look and feel through the PowerBuilder UI Theme feature
  - how to organize the menus in a modern way using the PowerBuilder RibbonBar control
- **Example Graph App** -- This demo app is newly added in 2019 R3. It uses the PowerBuilder WebBrowser control to render the third-party visual charts (such as Google Charts, Apache EChart etc.) which are exposed as JavaScript classes. These JavaScript/HTML5 charts can be served as a useful supplement to the PowerBuilder Graph DataWindow control.

  This app includes two application targets: one for Google Charts and the other for Apache EChart; and demonstrates

  - how to execute JavaScripts through the PowerBuilder WebBrowser control to generate a chart with different styles and adjust data display dynamically
  - how to connect the chart event with the WebBrowser control event so that JavaScript and PowerScript can interact with each other

  This demo app includes a readme file which explains in detailed steps how to incorporate Google Charts and Apache EChart (other JavaScript/HTML5 charts likewise) into your PowerBuilder application.

1.6 PowerBuilder IDE enhancements

PowerBuilder IDE has been enhanced with the following features:

- Double-clicking an object (such as applications, DataWindow object, windows, menus, user objects, functions, etc.) in the Browser window can open the painter associated with the object. Previously, you have to right-click the object and then select Edit to open the painter.
- When a DataWindow column is set to the DropDownDW edit style and associated with a DataWindow object, you can right click the DataWindow column and then select **Modify DropDownDW...** to directly open and edit the associated DataWindow object in the painter.
- The PowerBuilder system dialogs (such as Application, System Options, the various objects' Options dialogs etc.) have their layouts re-arranged, and the dialog size can be adjusted by dragging.
- The Enabled property of the Text control in the Header band of a DataWindow will be selected by default, and the tab order of the Text control will be set to 0 by default. Previously, the Enabled property is not selected and no tab order is set by default.

1.7 RibbonBar enhancements

You can now create a PBR (PowerBuilder resource) file to list the resource names (such as BMP, PNG etc.) used by the RibbonBar control and compile the PBR file with the application executable.

1.8 PBU conversion supports long data type

The PBU conversion functions (UnitsToPixels and PixelsToUnits) support a new data type (long) for the first argument. The value of the first argument can be an integer type or a long type. If the value of the first argument is a numeric value, then the return value is a long type by default.

1.9 WebBrowser enhancements

The WebBrowser control supports executing JavaScript by the following functions and events:

- EvaluateJavascriptAsync function -- Executes JavaScript asynchronously. This function triggers the EvaluateJavascriptFinished event.

  ```
  EvaluateJavascriptAsync (string javascript)
  ```

- EvaluateJavascriptSync function -- Executes JavaScript synchronously.

  ```
  EvaluateJavascriptSync (string javascript{, ref string result{, ref string
   error}})
  ```

- EvaluateJavascriptFinished event -- Triggered after the EvaluateJavascriptAsync function is executed.

  ```
  EvaluateJavascriptFinished (string result, string error)
  ```

- RegisterEvent function -- Registers the user defined event so that it can be triggered in JavaScript.

```
RegisterEvent (string eventname)
```

PowerBuilder user events can be defined with one string-type parameter and string-type return value. For more, see Section 4.5.3, "Defining user events for WebBrowser".

- UnregisterEvent function -- Unregisters the user defined event that is registered using the RegisterEvent function.

```
UnregisterEvent (string eventname)
```

The WebBrowser control supports basic and digest authentications:

- When the web page to be accessed needs basic or digest authentication, the WebBrowser control will automatically display a login window for the user to enter the user name and password for authentication. If the authentication fails, this window will display again until the authentication is successful or the authentication operation is cancelled.

The WebBrowserSet and WebBrowserGet functions are enhanced to support to access the local files and resources:

- allow-file-access-from-files -- Whether to allow access to the local files (XML etc.).
- enable-media-stream -- Whether to allow access to the microphone or camera.

1.10 JSONParser and JSONGenerator enhancements

In order to support

1. getting and setting an item by path, and
2. handling special values such as JsonNaN (invalid number), JsonPositiveInfinity (positive infinity), and JsonNegativeInfinity (negative infinity),

JSONParser object has added or improved the following functions:

- New GetNumberType function -- Determines the value type of a number item: JsonNumber, JsonNaN, JsonPositiveInfinity, or JsonNegativeInfinity.
- The following functions have been enhanced to support JsonNaN, JsonPositiveInfinity, and JsonNegativeInfinity:
  - LoadString supports to load a string which contains JsonNaN/JsonPositiveInfinity/JsonNegativeInfinity value
  - LoadFile supports to load a file which contains JsonNaN/JsonPositiveInfinity/JsonNegativeInfinity value
  - GetItemObjectJSONString and GetItemArrayJSONString support the returned object or array which contains JsonNaN/JsonPositiveInfinity/JsonNegativeInfinity value
  - GetItemNumber supports to get value which is JsonNaN/JsonPositiveInfinity/JsonNegativeInfinity

- New GetItemByPath function -- Gets the item handle according to the path.
- New ContainsPath function -- Checks if a path exists.
- New GetItemObjectJSONString and GetItemArrayJSONString functions -- Gets the string value (in JSON format) of the object item or array item.
- The following functions have been enhanced to get items by path: GetChildCount, GetChildKey, GetChildItem, GetItemArray, GetItemBlob, GetItemBoolean, GetItemDate, GetItemDateTime, GetItemNumber, GetItemObject, GetItemString, GetItemTime, GetItemType, and ContainsKey.

JSONGenerator object has added or improved the following functions:

- New ImportFile and ImportString functions -- Imports a JSON item from a JSON string or file.
- New GetItemByPath and GetPathByItem functions -- Gets item handle by path or gets path by item handle.
- The following functions have been enhanced to support JsonNaN, JsonPositiveInfinity, and JsonNegativeInfinity:
  - AddItemNumber, ImportFile and ImportString support the parameter which contains JsonNaN/JsonPositiveInfinity/JsonNegativeInfinity value
  - GetJSONString supports the returned value which contains JsonNaN/ JsonPositiveInfinity/JsonNegativeInfinity value
- The following functions have been enhanced to add items by path: AddItemArray, AddItemBlob, AddItemBoolean, AddItemDate, AddItemDateTime, AddItemNull, AddItemNumber, AddItemObject, AddItemString, and AddItemTime.

1.11 HTTPClient enhancements

The HTTPClient object has added the following new properties:

- IgnoreServerCertificate -- Ignores certain type(s) of server certificate error when sending a request.
- CheckForServerCertRevocation -- Checks if the server certificate is revoked when sending a request.

1.12 64-bit rich text editor support

PowerBuilder 2019 R3 supports the 64-bit version of the TX Text Control. The user can select "Built-in TX Text Control ActiveX 28.0" (instead of Microsoft RichEdit Control) as the 64-bit editor.

For details, refer to Section 4.5.1.3, “Rich text editors” in *Application Techniques*.

1.13 UI accessibility and automation support

PowerBuilder supports the following two technologies of Windows accessibility and automation -- Microsoft Active Accessibility (MSAA) and Microsoft UI Automation.

MSAA is a legacy technology introduced in Windows 95; it supports the PowerBuilder standard controls well; but it imposes major limitations on the PowerBuilder custom controls such as DataWindows. Starting from PowerBuilder 2019 R3, PowerBuilder supports a newer and more capable technology which overcomes the limitations of MSAA; this new technology called Microsoft UI Automation offers a richer set of properties and extended interfaces to manipulate not only standard controls but also custom controls (such as PowerBuilder DataWindows and child controls in DataWindows).

With the Microsoft UI Automation technology, the UI elements of the PowerBuilder control can be accessible to the screen reader (such as Windows Narrator), the accessibility tool (such as Accessibility Insights, Inspect), and the automated testing tool (such as QTP) in Windows 10 and Windows Server 2019.

To enable support for PowerBuilder controls/objects through Microsoft UI Automation, you will need to set the AccessibleName and AccessibleDescription properties in the controls' Property tab page.

The Microsoft UI Automation technology allows you not only to access and test the PowerBuilder standard controls (see Table 1), but also the PowerBuilder DataWindow controls (see Table 2 and 3).

Table 1: PowerBuilder controls/objects **supported** by Microsoft UI Automation

Table 1.2:

Animation	GroupBox	Picture	StaticHyperText
CheckBox	HProgressBar	PictureButton	Tab
CommandButton	HScrollBar	PictureHyperText	TreeView
DataWindow	HTrackBar	PictureListBox	UserObject
DatePicker	ListBox	RadioButton	VProgressBar
DropDownListBox	ListView	SingleLineEdit	VScrollBar
DropDownPictureListBox	MonthCalendar	StaticText	VTrackBar
EditMask	MultiLineEdit		

The following PowerBuilder controls/objects are **unsupported** by Microsoft UI Automation: Graph, InkEdit, InkPicture, Line, OLE, Oval, Rectangle, RibbonBar, RoundRectangle, RichTextEdit, and WebBrowser.

Table 2: DataWindow presentation styles **supported** by Microsoft UI Automation

Table 1.3:

CrossTab	Label
FreeForm	N-Up
Grid	Tabular

Group	TreeView

DataWindow controls of the following presentation styles are **unsupported**: Graph, OLE 2.0, Composite, and RichText.

Table 3: Controls in DataWindow **supported** by Microsoft UI Automation

Table 1.4:

Button Column (and the following edit styles are supported: CheckBox, DropDownListBox, DropDownDataWindow, Edit box, EditMask, and RadioButtons)	Computed Field GroupBox Text

The following controls in DataWindow are **unsupported** by Microsoft UI Automation: Graph, InkPicture, Line, OLE Object, Oval, Rectangle, RoundRectangle, Picture, Report, and TableBlob.

Examples

The following statements set the AccessibleName and AccessibleDescription properties for a command button in a Window:

```
cb_1.accessiblename = "Delete"
cb_1.accessibledescription = "Deletes selected text"
```

The following statement sets the AccessibleName property of a button in a DataWindow object:

```
dw_1.Object.b_1.accessiblename = "Update"
```

Deployment

When you deploy an accessible application, you must deploy the pbacc.dll and PBAccessibility.dll files.

1.14 PostgreSQL enhancements

The following enhancements have been made to the PostgreSQL database-related tasks in the database painter:

- You can now use the "Show System Procedures" context menu to show or hide the system procedures for PostgreSQL in the Objects view.

1.15 Microsoft SQL Server enhancements

The following enhancements have been made to the Microsoft SQL Server database-related features:

- The Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) is supported. You can select the **MSOLEDBSQL SQL Server** interface from Database Profiles to connect to the Microsoft SQL Server 2012, 2014, 2016, 2017, or 2019 databases using the Microsoft OLE DB Driver for SQL Server. The **MSOLEDBSQL SQL Server** interface supports exactly the same features as the SNC SQL Native Client interface does. The additional new features of Microsoft OLE DB Driver (compared to SQL Native Client) will be supported in the next PowerBuilder version.

The Microsoft OLE DB Driver for SQL Server 18.2 or later is supported. For more, see Section 3.4, "Using Microsoft SQL Server" in *Connecting to Your Database*.

- The PowerBuilder application can connect with Microsoft SQL Server database through TLS 1.2 (using either MSOLEDBSQL SQL Server or SNC SQL Native Client interface).

1.16 Installer enhancements

PowerBuilder Installer has the following changes or enhancements:

- You can select to use the PostgreSQL engine (in addition to the SQL Anywhere engine) for running the PowerBuilder demo database.

  If you specify the PostgreSQL engine, make sure both the PostgreSQL client (included in Command Line Tools of the PostgreSQL Windows installer) and PostgreSQL ODBC driver (32-bit) are installed.

- You can install PowerBuilder Runtime as a separate component for PowerBuilder IDE.

  PowerBuilder Runtime must be installed before PowerBuilder IDE if you want to install PowerBuilder Runtime and PowerBuilder IDE sequentially one by one. For more information, refer to Section 9.2.4.2, "Installing PowerBuilder Runtime" in *Application Techniques*.

- You can install PowerBuilder Compiler from PowerBuilder Installer, now that PowerBuilder Compiler installation program is integrated into PowerBuilder Installer.

  For more information, refer to Section 8.2.1, "Installing PowerBuilder Compiler" in *Users Guide*.

- You can now modify installation if you have not fully installed all the products in a certain version. The Install button on the entry page of the installer will change to Modify.

1.17 App upgrade notes

This section only highlights a few important notes for application upgrade. For more information, refer to Upgrading PowerBuilder Applications.

For upgrading from Beta to GA

If you want to upgrade from Beta1 or Beta2 to GA, make sure to upgrade both IDE and runtime and use the runtime installed from GA, as GA has moved the third-party DLL files in "ThirdPartyLegal\MSRedist\" from IDE to Runtime, if you want to continue using the runtime installed from beta, you will need to manually copy the DLL files from ThirdPartyLegal\MSRedist\x86 or ThirdPartyLegal\MSRedist\x64 to the runtime folder %AppeonInstallPath%\Common\PowerBuilder\Runtime [version] or %AppeonInstallPath%\Common\PowerBuilder\Runtime [version]\x64, otherwise the IDE installed from GA will not work with the runtime installed from beta.

If you want to upgrade from Beta1 or Beta2 to GA, please 1) uninstall **PowerClient App Publisher** from the server, and uninstall the launcher and service from the client, including

PowerClient Launcher (or **Cloud App Launcher**), **PowerClient Service**, and **PowerClient App Shell**, as GA has re-named these programs; 2) undeploy/uninstall the app from the server and the client; 3) configure and deploy a new PowerClient project in GA, as GA has changed the project settings. The app deployed with beta will no longer work with the GA version.

For upgrading PowerBuilder projects

For upgrading PowerBuilder projects from 2019/2019 R2 to 2019 R3, refer to https://docs.appeon.com/pb2019r3/upgrading_pb_apps/ch04.html.

1.18 Obsolete features

Obsolete features are still available to use, but are no longer eligible for technical support and will no longer be enhanced.

For ***migration path*** of these obsolete features, refer to Obsolete/Discontinued Features in Product Availability Matrix web page.

The following features are considered as obsolete features, starting from version 2019 R3:

- Built-in Rich Edit Control (TE Edit Control)
- Inet object
- Opening Internet Explorer browser pages in the OLEControl (GetURL, PostURL, HyperlinkToURL)
- Windows 7 operating system

The following features are considered as obsolete features, starting from version 2019 R2:

- Docking windows
- Windows Server 2008 operating system

The following features are considered as obsolete features, starting from version 2019:

- .NET Web Service target (using SOAP)
- .NET Assembly target

The following features are considered as obsolete features, starting from version 2017 R3:

- Web Service proxy for connecting to SOAP server (including SoapConnection class, SoapException class, SoapPBCookie class, UDDIProxy class)
- DataWindow data source using OData Service and SOAP Web Service
- Web DataWindow
- EJB client

2 New Features in PowerBuilder 2019 R2

About this chapter

This chapter introduces the new features in PowerBuilder 2019 R2.

2.1 Updated system requirements

PowerBuilder 2019 R2 supports the following system:

- Windows Server 2019
- Oracle 18c and 19c
- PostgreSQL 11 and 12
- SQL Server 2019

2.2 New RibbonBar Control

The RibbonBar control enables you to create ribbons which are a modern way of organizing user commands in user interfaces. It is composed of Category, Panel, Group, LargeButton (with or without RibbonMenu), SmallButton (with or without RibbonMenu), CheckBox, ComboBox, TabButton (with or without RibbonMenu), and Spin (currently unsupported).

To help developers create a ribbon efficiently, a tool called "RibbonBar Builder" is provided. The RibbonBar Builder allows developer to modify the RibbonBar control template in an XML editor while preview its graphical UI on the fly.

For more details, refer to Section 2.90, “RibbonBar control” in *Objects and Controls* and Section 4.4, “Working with RibbonBar” in *Users Guide*.

2.3 Calling .NET Assembly

PowerBuilder applications can directly call .NET assemblies, via two objects: DotNetAssembly and DotNetObject. DotNetAssembly is used to load the .NET assembly, and DotNetObject is used to map to the .NET class in the assembly. After you create DotNetAssembly and DotNetObject instances in PowerBuilder, you can directly call the functions defined in the .NET class.

To help developers write scripts in a more productive way, a tool called ".NET DLL Importer" is provided to help developers write scripts to correctly call functions in the .NET class. .NET DLL Importer can import the names and data types of the .NET classes, functions, properties, and parameters from the .NET assembly to the application PBL. It creates the DotNetObject object as an NVO for each .NET class and then imports the .NET functions to the NVO. After that the developer can write scripts to call the NVO and function directly to execute the corresponding .NET code.

For more details, refer to Section 5.1, “Calling .NET Assembly in an Application” in *Application Techniques*, Section 2.16, “DotNetAssembly object” in *Objects and Controls*, and Section 2.17, “DotNetObject object” in *Objects and Controls*.

2.4 Enhanced UI Theme

Each individual control, object, user object, or window can have its own theme settings now, for example, one button can have theme settings different from the other buttons. And controls of the same type in the same window can have their own theme settings, for example, all group boxes in one window can have theme settings different from the group boxes in the other window. To learn how to configure the theme settings for a control or object, refer to the section called “Custom themes” in *Users Guide*.

Besides that, the following UI elements can be set by the theme file:

- The UI settings of custom visual user objects
- The background color, title bar, border, and system buttons (such as maximize, minimize and restore buttons) of window and user object
- The scroll bar on the OLE control, user object, and window
- The menu, toolbar, and status bar of window and user object

2.5 New WebBrowser Control

The WebBrowser control can be used to create a Web browser that:

- Supports browsing web page that contains JavaScript
- Supports browsing HTML and HTML5 pages
- Supports browsing videos at common formats in web page
- Supports playing flash (with the flash plug-in installed by the user)
- Supports printing web pages as PDFs and responding to the print events
- Supports browsing web pages that support multiple languages
- Supports the various HTTPS protocols
- Supports right-mouse button context menus
- Supports browsing a PDF file online
- Supports web page zoom in/out
- Supports browsing local files (including htm, gif, jpg, jpeg, png, swf, txt, c, cpp, pdf, etc.)
- Supports responding to the error event from the server certificate
- Supports dynamically configuring the file download location and proxy settings.

For more about the supported/unsupported features and the properties/functions/events of the WebBrowser object, refer to Section 2.150, “WebBrowser control” in *Objects and Controls*.

2.6 Enhanced CoderObject and CrypterObject

The following new functions are added for the CoderObject object:

- Base32Decode -- Decodes a string value using Base32 decoder.
- Base32Encode -- Encodes a blob value using Base32 encoder.
- Base64UrlDecode -- Decodes a string value using Base64Url decoder.
- Base64UrlEncode -- Encodes a blob value using Base64Url encoder.

The following new functions are added for the CrypterObject object:

- SymmetricGenerateKey -- Generates a secret key for asymmetric algorithm.

For more information, refer to Section 2.7, “CoderObject object” in *Objects and Controls* and Section 2.15, “CrypterObject object” in *Objects and Controls*.

2.7 Enhanced ExtractorObject

The following new functions are added for the ExtractorObject object:

- GetFilesCount -- Gets the number of files contained in the archive.
- GetFilesList -- Gets the list of files in the compressed package.
- Extract -- Extracts only the files in the specified package, or extracts the specified file in the compressed package into the buffer.

For more information, refer to Section 2.33, “ExtractorObject object” in *Objects and Controls*.

2.8 Enhanced Source Control

Source control has the following enhancements:

- (SVN and Git) Supports to view differences from the SVN/Git Commit (SVN Get/Release Lock, Revert, or Resolve) dialog -- In the Commit (SVN Get/Release Lock, Revert, or Resolve) dialog, double click an item from the object list to compare the local object with the version last sync with source control and view the differences.
- (SVN and Git) Supports Application Properties in source control -- Application Properties (such as theme settings, rich text settings) can be manipulated by the SVN/Git source control.
- (SVN only) Supports the svn:needs-lock property -- PowerBuilder IDE provides no direct option for setting this property; you can set this property using an SVN client (such as TortoiseSVN), and after that, you can manipulate (such as lock, commit etc.) the object which has the svn:needs-lock property in the PowerBuilder IDE.
- (SVN only) Supports using multi-languages in Repository URL, Workspace File, User ID etc. when connecting with, uploading to, or downloading from the source control server.

- (SVN only) Enhances Get Lock to allow users to update the object first before locking the object if newer version of the object exists on the source code server.

- (Git only) Supports creating and switching to the branch -- PowerBuilder IDE provides no direct option for creating or switching to a branch; but you can create and switch to the branch using a third-party tool (such as TortoiseGit), and then restart PowerBuilder IDE to automatically switch to the branch; and after that you can manipulate the objects under the branch in the PowerBuilder IDE. For detailed steps, see Section 1.3.3.11, “Use branches” in *Users Guide*.

2.9 Using NativePDF as Default

For new DataWindows, the NativePDF! (using PDFLib) method is automatically selected by default when you select Save Rows As from the File menu in the DataWindow painter and select PDF as the file type, or when you use the SaveAs method with PDF! as the file type.

But for existing DataWindows which are kept the same as before to use the Distill! method with Ghostscript as the default method, if you want to save them to PDF using PDFlib, you can select the "Always use NativePDF! method for PDF export" in the Application properties dialog box. This option is effective to all DataWindows in the application. For more, see Section 6.2.3.1.1, “Saving as PDF using NativePDF! method with PDFlib” in *Users Guide*.

The two INI settings NativePDF_IncludeCustomFont and NativePDF_Valid will no longer take effect.

Besides that, you are able to set a few properties for the generated PDF file, such as the application name, author, subject, and keywords.

2.10 Testing DataWindows using AscentialTest

You can automate the testing of PowerBuilder DataWindow objects by using AscentialTest 9.5.2 or above. Please visit this page provided by AscentialTest that explains how automated tests are written for the PowerBuilder DataWindow objects.

2.11 PowerBuilder IDE enhancements

Other new or changed features related with PowerBuilder IDE are as below:

- New picture selection dialog -- Previously the picture is selected from a dropdown list; now the picture is selected from a dialog box. In the dialog box, you can select from the custom image or the built-in image; the built-in images are grouped by categories, and can be filtered by size or searched by name.

  Some images which have a random number at the end of their names have been renamed, for example, ArrangeTables5! is renamed to ArrangeTables2!, but ArrangeTables5! is preserved so if it is referenced in your application, it will still show under preview and at runtime like before, but in the dialog you can only select ArrangeTables2! (and cannot select ArrangeTables5! any more).

- Enhances Browser window for searching object -- A filter box is added to the Browser window which allows users to input the object name and display only the objects that match in each tab. The total amount of search results will be displayed at the bottom right corner of the window.

- Adds Open Containing Folder menu for workspace, target and library -- The Open Containing Folder menu is available when you right click on the workspace, target, or library in the System Tree or Library painter.
- **C# Model Generator** has been moved out of the PowerBuilder IDE, as an independent plug-in. It is renamed as "DataWindow Converter" and installed as a plug-in from the SnapDevelop installation program.
- C# projects can no longer be loaded into the PowerBuilder IDE. You can open the C# projects in C# editor such as SnapDevelop, Visual Studio, etc.

2.12 PowerBuilder runtime enhancements

All PowerBuilder runtime DLLs have been digitally signed now.

2.13 Other New Features or Changes

Other new features related with PowerScripts or PowerBuilder objects are as below:

- Supports Placeholder property for SingleLineEdit control -- The Placeholder property specifies a short description for the expected value of the input field. The Placeholder value can be used as a text label or hint; hence can help to reduce the number of UI controls. The Placeholder value will not be displayed, 1) if the Text property is set (the Text value will be displayed first); or 2) when the input field has focus.
- DataWindow CheckBox and RadioButton edit styles have enhanced vertical centering of the box/circle and the text at runtime (the box/circle is still aligned to the top at design time).
- Enhances Bitmap DataWindow expression function to validate and display the image file if the image file has no file extension.
- The RichText DataWindow that uses the TX Text Control can now directly generate PDFs using the DataWindow SaveAs function, for example, dw_1.SaveAs ("c:\dw_one.pdf", PDF!, false).

Other new or changed features related with PowerBuilder/InfoMaker Installer:

- Starting from 2019 R2, a user can select an MR version and complete in one step the installation of both the full version and the selected MR.

3 New Features in PowerBuilder 2019

About this chapter

This chapter introduces the new features in PowerBuilder 2019.

3.1 Empowering PowerBuilder with C# features

.NET Packages

Two .NET packages (**PowerBuilder.Data** and **PowerBuilder.Data.AspNetCore**) are provided to implement the .NET replacement for the PowerBuilder DataStore. **PowerBuilder.Data** provides a pure .NET DataStore compatible with the .NET Core along with related libraries to integrate with PowerBuilder DataWindows or DataStores through a REST interface. The .NET DataStore works similar to the PowerBuilder DataStore and maintains same naming convention of its APIs for easy porting of existing project assets. **PowerBuilder.Data** also provides a ModelStore object which works against the .NET data model and can be used to replace the .NET DataStore.

Also, you can enhance your .NET projects with a new .NET ORM framework (called **SnapObjects**), which is also compatible with the .NET Core just like **PowerBuilder.Data** and **PowerBuilder.Data.AspNetCore**.

For more information about the PowerBuilder .NET APIs and SnapObjects .NET APIs, refer to the Documentation Center.

C# Migration

A batch DataWindow/DataStore conversion utility called **C# Model Generator** is provided to generate the C# data objects and models for the .NET DataStore and ModelStore.

Also provided is a SqlExecutorExtention object in **PowerBuilder.Data.AspNetCore**, which can be used to virtually copy n' paste embedded SQL from PowerScript projects to new C# projects.

C# IDE

A relatively full-featured C# IDE is provided on a standalone basis (called **SnapDevelop**) and can be launched from the PowerBuilder IDE. It supports development of non-visual projects, such as C# Web APIs, C# non-visual assemblies, and unit testing (with xUnit). And it provides powerful developer productivity tools, such as project wizards, advanced auto-scripting, and C# language services.

For more information about SnapDevelop, refer to the Documentation Center.

3.2 New UI theme

A new UI theming system is provided to allow for codeless approach to how your application UI is rendered. For detailed instructions on how to use the new UI theme, refer to Section 2.1.4.5, “Specifying a theme for the application UI” in *Users Guide*.

System themes and custom themes

Four new system themes (Flat Design Blue, Flat Design Dark, Flat Design Grey, and Flat Design Silver) are provided for you to apply to the window, DataWindow, and all visual controls (except Line, Oval, Rectangle, RoundRectangle, Picture, PictureHyperLink, and

Animation) in your applications. You can also customize these system themes or create your own themes based on these system themes.

Applying a theme

To apply a theme to an application, you can either set it in the Themes tab in the Additional Properties of an application object, or use the ApplyTheme function to set the theme dynamically.

The theme will work in runtime, and has no effect in design time.

Modifying the settings of a theme

If after applying a theme, you want to further adjust the display of certain controls/states, you can open the "theme.json" file of the theme in the default or specified directory, and change the corresponding theme settings (with caution).

In case you want to restore the settings of a system theme to its original state, you can do it using the Restore button provided in the Themes tab in the Additional Properties of an application object. The Restore button is only effective when the system theme is located in the default directory.

What can be set by a theme

Due to technical difficulty, a few controls and their states won't reach desired UI effects even through you apply a theme. For more information on what can be set by a theme and what cannot, refer to Section 2.1.4.5, "Specifying a theme for the application UI" in *Users Guide*.

3.3 Incorporating TX Text Control as built-in editor

Starting from PowerBuilder 2019, a special OEM version of TX Text Control ActiveX is incorporated as a built-in rich text editor in PowerBuilder. This is provided in all editions of PowerBuilder at no additional cost, and is highly recommended to be used for backwards compatibility reasons by all existing PowerBuilder projects that already make use of the RichTextEdit of SAP PowerBuilder version 12.6 or older.

To select to use the OEM version of TX Text Control: in the Application properties dialog box, select the RichTextEdit Control tab, and then select the first option "Built-in TX Text Control".

3.4 New or enhanced PowerBuilder objects

3.4.1 Enhanced RESTClient object

The following functions are added to the RESTClient object:

- Submit can not only send data from the application client to the RESTful web service but also get the response body from the RESTful web service.

  The data to be submitted may be from either a DataWindow, DataStore, DataWindowChild, or JSONPackage. You can specify one or multiple DataWindow buffers, and also the range in the DataWindow, from which to submit the data. The request supports OAuth 2.0 authorization.

  ```
  objectname.Submit(string urlName, ref string response, DWControl dwObject{,
   boolean format})
  ```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject
 {,DWBuffer dwbuffer}, boolean changedonly, boolean format)
```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject,
 boolean primarydata, boolean filterdata, boolean deletedata, boolean dwcdata {,
 boolean format})
```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject,
 DWBuffer dwbuffer{,long startrow{, long endrow{, long startcol{, long endcol}}}}
 {, boolean format})
```

```
objectname.Submit(string urlName, ref string response, ref JsonPackage package)
```

- SendDeleteRequest: Sends the HTTP DELETE request to the server and then gets the content of the server response.

  Previously only the HTTPClient object supports sending a request to the RESTful web service. Now you can directly send a request from the RESTClient object, and the request supports OAuth 2.0 authorization.

  ```
  SendDeleteRequest(string urlName{, string data }, ref string response)
  ```

- SendGetRequest: Sends the HTTP GET request to the server and then gets the content of the server response.

  ```
  SendGetRequest(string urlName, ref string response)
  ```

- SendPatchRequest: Sends the HTTP PATCH request to the server and then gets the content of the server response.

  ```
  SendPatchRequest(string urlName, string data, ref string response)
  ```

- SendPostRequest: Sends the HTTP POST request to the server and then gets the content of the server response.

  ```
  SendPostRequest(string urlName, string data, ref string response)
  ```

- SendPutRequest: Sends the HTTP PUT request to the server and then gets the content of the server response.

  ```
  SendPutRequest(string urlName, string data, ref string response)
  ```

- GetJWTToken: Gets the JWT token using the POST method.

  ```
  GetJWTToken (string urlName, string data, ref string token)
  ```

- SetJWTToken: Sets the JWT token string to the HTTP request header which will be sent to the server.

  ```
  SetJWTToken(string jwtToken)
  ```

- GetOAuthToken: Gets the OAuth 2.0 access token.

  ```
  GetOAuthToken (TokenRequest tokenRequest, ref string token)
  ```

- SetOAuthToken: Sets the OAuth 2.0 token string to the HTTP request header which will be sent to the server.

  ```
  SetOAuthToken(string token)
  ```

- RetrieveOne: Retrieves one data row to the DataWindow, DataWindowChild, or DataStore from the RESTFul Web service.

  ```
  RetrieveOne (DWControl dwObject, string urlName {,string data})
  ```

The following functions for the RESTClient object are modified:

- Retrieve - Retrieves data to the DataWindow, DataWindowChild, or DataStore from the RESTFul Web service.

  If the data received from the RESTful web service is compressed as gzip, it will be automatically decompressed. Only gzip compression format is supported at this moment. You can use the SetRequestHeader function to set the Accept-Encoding header to allow only the gzip compression format.

- SetRequestHeader - Supports to add a request header or add/replace the value in the existing request header if the header already exists.

  ```
  SetRequestHeader ( string headerName, string headerValue{, Boolean replace } )
  ```

For more details about these functions, refer to Section 2.87, “RESTClient object” in *Objects and Controls*.

3.4.2 Enhanced HTTPClient object

The following functions for the HTTPClient object are enhanced:

- SetRequestHeader: supports to add a request header or add/replace the value in the existing request header if the header already exists.

  ```
  SetRequestHeader ( string headerName, string headerValue{, Boolean replace } )
  ```

- SendRequest: supports to encode the data with the charset which is specified by the user in the Content-Type request header, if charset is not specified, this function will encode the data in UTF-8 by default.

- GetResponseBody: supports to encode the data with the charset which is specified by the user in the Content-Type request header; if charset is not specified, this function determines the encoding type based on the BOM header, and then converts the data into UNICODE.

For more details about these functions, refer to Section 2.41, “HTTPClient object” in *Objects and Controls*.

3.4.3 Enhanced JSONPackage object

The GetValue function of the JSONPackage object always returns the result in string, and the SetValue function only sets string-type values. Now there are more functions in the JSONPackage object for getting or setting values of various types.

- GetValueBlob -- Gets the blob value of the key.

- GetValueBoolean -- Gets the boolean value of the key.

- GetValueDate -- Gets the date value of the key.
- GetValueDateTime -- Gets the datetime value of the key.
- GetValueNumber -- Gets the number value of the key.
- GetValueString -- Gets the string value of the key.
- GetValueTime -- Gets the time value of the key.
- SetValueBlob -- Sets the blob value for a key.
- SetValueBoolean -- Sets the boolean value for a key.
- SetValueDate -- Sets the date value for a key.
- SetValueDateTime -- Sets the datetime value for a key.
- SetValueNumber -- Sets the number value for a key.
- SetValueString -- Sets the string value for a key.
- SetValueTime -- Sets the time value for a key.

You can now directly get values from the JSONPackage object into DataWindows, or set the key values in the JSONPackage object from DataWindows.

- GetValueToDataWindow -- Gets the value of the key and inserts it into a DataWindow control, DataStore object, or DataWindowChild object.
- SetValueByDataWindow -- Sets the value of the key using the data from a DataWindow control, DataStore object, or DataWindowChild object.

The JSONPackage object also provides the GetItemType function which works the same as the GetItemType function in the JSONParser object.

- GetItemType -- Gets the type of item.

The SaveToFile and GetJsonBlob functions for JSONPackage (and JSONGenerator) are enhanced to specify the character encoding of the resulting blob.

- SaveToFile

  ```
  SaveToFile ( FileName {, Encoding e} )
  ```

- GetJsonBlob

  ```
  GetJsonBlob ( {Encoding e} )
  ```

For the newly-added or enhanced functions, see Section 2.47, “JSONPackage object” in *Objects and Controls* for more details.

The following new property is added to the JSONPackage object (also added to JSONParser):

- ReturnsNullWhenError -- Specifies whether the getting value function returns a null value when error occurs.

You can use this property to avoid throwing an exception in cases when a getting item function returns null.

See Section 3.237, “ReturnsNullWhenError” in *Objects and Controls* for more details.

3.4.4 Enhanced JSONGenerator

The SaveToFile and GetJsonBlob functions for JSONGenerator (and JSONPackage) are enhanced to specify the character encoding of the resulting blob.

- SaveToFile

  ```
  SaveToFile ( FileName {, Encoding e} )
  ```

- GetJsonBlob

  ```
  GetJsonBlob ( {Encoding e} )
  ```

For more information, see Section 2.4.664, “SaveToFile” in *PowerScript Reference* and Section 2.4.286, “GetJsonBlob” in *PowerScript Reference*.

3.4.5 Enhanced JSONParser object

The following new function is added to the JSONParser object:

- ContainsKey -- Checks if the key name exists.

  You can use this function to check whether certain key exists in a JSONParser object before executing other functions, such as GetItem.

  See Section 2.4.91, “ContainsKey” in *PowerScript Reference* for more details.

The following function is enhanced:

- GetItemType -- Gets the type of item.

  It is now possible to specify the key of a child item, and directly get the type of the child item.

  See Section 2.4.285, “GetItemType” in *PowerScript Reference* for more details.

The following new property is added to the JSONParser object (also added to JSONPackage):

- ReturnsNullWhenError -- Specifies whether the getting item function returns a null value when error occurs.

  You can use this property to avoid throwing an exception in cases when a getting item function returns null.

  See Section 3.237, “ReturnsNullWhenError” in *Objects and Controls* for more details.

3.4.6 New CompressorObject and ExtractorObject objects

Two new objects called CompressorObject and ExtractorObject are added to compress and decompress the folder or file(s), or the byte data stream. The supported compression formats include ZIP, 7ZIP, GZIP and TAR, and the ZIP and 7ZIP formats support AEM-256 encryption for password; the supported extraction formats include ZIP, 7ZIP, RAR, GZIP, TAR, LZMA, and LZMA86.

For more information about these objects, refer to Section 2.9, "CompressorObject object" in *Objects and Controls* and Section 2.33, "ExtractorObject object" in *Objects and Controls*.

For syntax of compressing files or data stream, refer to Section 2.4.85, "Compress" in *PowerScript Reference*; for syntax of extracting a compressed archive or data stream, refer to Section 2.4.165, "Extract" in *PowerScript Reference*; for code examples on compressing and extracting files with the HTTPClient object, the RESTClient object, or the OAuthClient object, refer to Section 4.7.3, "Compressing and extracting data" in *Application Techniques*.

3.4.7 New Import & Export JSON functions

The following functions are added to import/export a single data row between a JSON string and a DataWindow control, DataStore object, or DataWindowChild object:

- ImportRowFromJson: Inserts a data row from a JSON string into a DataWindow control, DataStore object, or DataWindowChild object.

  ```
  ImportRowFromJson( string json, long row {, ref string error} {, DWBuffer
   dwbuffer})
  ```

- ExportRowAsJson: Exports a data row from a DataWindow control, DataStore object, or DataWindowChild object to the JSON string.

  ```
  ExportRowAsJson (long row {, DWBuffer dwbuffer})
  ```

For more information, see Section 9.99, "ImportRowFromJson" in *DataWindow Reference* and Section 9.30, "ExportRowAsJson" in *DataWindow Reference*.

3.4.8 New JSON format

The SnapObjects ModelStore can exchange data with PowerBuilder DataWindow in JSON strings, so now the following JSON formats are supported:

- Plain JSON (formerly called simple JSON)
- DataWindow JSON (formerly called standard JSON)
- ModelStore JSON (added in version 2019) **Note: removed in version 2019 R2**

For more about these formats, refer to Section 4.7.1, "Supported JSON formats" in *Application Techniques*.

3.5 New Windows 10 style icons and small pictures

A set of Windows 10 style icons are provided for selection under the "icons" list and the "small pictures" list in the Property tab for controls. The developer can quickly find out these new Windows 10 icons, according to the text appended to the icon name, for example, "_icon_2" is appended to the icon name, and "_2" is appended to the small picture name.

3.6 64-bit enhancements

PowerBuilder 2019 has enhanced support for 64-bit, so the following bugs no longer exist:

1. Previously, when developers used the registry functions (including RegistryDelete, RegistryKeys, RegistryGet, RegistrySet, & RegistryValues) to access the 64-bit registry entries in a 64-bit operating system, they were incorrectly redirected to the Wow6432Node registry entry; and now they can access the correct registry entry.

   And to support this, a longlong type enumeration value (RegLongLong!) is added for the RegistryGet and RegistrySet functions. For more information, see Section 2.4.617, “RegistryGet” in *PowerScript Reference* or Section 2.4.619, “RegistrySet” in *PowerScript Reference*.

2. Previously, 32-bit PowerBuilder application and 64-bit PowerBuilder application cannot coexist on the same client machine because they used the same location to store runtime files; and now they use different locations, so they can co-exist on the same machine.

3.7 PBC enhancements

PBC (PowerBuilder Compiler) supports a new parameter "/pd" which can specify whether to generate the PBD/DLL file for a PBL.

3.8 New online installer -- Appeon Installer

Starting from 2019, a new and more efficient way to install Appeon products is introduced -- Appeon Installer, which is an online installer with a self-extracting download that leads you through the installation process. The machine will be required to connect with Internet during the installation process.

For detailed instructions on using Appeon Installer, refer to Section 3.1, “Online Installation” in *Installation Guide*.

3.9 Classified features

While new features are added in PowerBuilder, some of the existing features might no longer be needed or recommended for use. You will find the following three classifications of features in the PowerBuilder Help.

- **Discontinued** - feature that has been completely removed from the product.

  For example, EAServer projects/targets, PowerBuilder .NET IDE, and Windows Form projects/targets are discontinued features.

- **Obsolete** - feature that is available, but is no longer eligible for technical support and will no longer be enhanced.

  For example, Web DataWindow, SOAP client, DataWindow data source using OData Service and SOAP Web Service, .NET Web service target, .NET assembly target are obsolete features.

- **Stable** – feature that is available and is still eligible for technical support, but will no longer be enhanced.

4 Bug Fixes & Known Issues in PowerBuilder 2019 R3

Refer to https://docs.appeon.com/pb2019r3/release_bulletin_for_pb/.

5 Bug Fixes & Known Issues in PowerBuilder 2019 R2

The bug fixes and known issues for PowerBuilder 2019 R2 are listed in the Release Bulletin for PowerBuilder at the following links:

Bug Fixes for PowerBuilder 2019 R2: https://docs.appeon.com/pb2019r2/release_bulletin_for_pb/bug_fixes.html

Known Issues for PowerBuilder 2019 R2: https://docs.appeon.com/pb2019r2/release_bulletin_for_pb/known_issues.html

6 Bug Fixes & Known Issues in PowerBuilder 2019

The bug fixes and known issues for PowerBuilder 2019 are listed in the Release Bulletin for PowerBuilder at the following links:

Bug Fixes for PowerBuilder 2019: https://docs.appeon.com/pb2019/release_bulletin_for_pb/bug_fixes.html

Known Issues for PowerBuilder 2019: https://docs.appeon.com/pb2019/release_bulletin_for_pb/known_issues.html