What's New

Appeon PowerBuilder® 2019 R3

Contents

1 PowerBuilder 2019 R3 の新機能

この章について

この章では、PowerBuilder 2019 R3 の新機能を紹介します。

1.1 ランタイムと IDE の分離

1.1.1 ランタイムと IDE を分離する理由

PowerBuilder ランタイムファイルは、独立したコンポーネント “PowerBuilder ランタイム” として PowerBuilder IDE から分離されました。

ランタイムと IDE の分離によりランタイムと IDE を別々にインストールおよびアップグレードできます。また、1 つの IDE とネイティブ C/S アプリケーションは、IDE をアップグレードしたりアプリケーションを再コンパイルすることなく、複数バージョンのランタイムで動作できます。たとえば、以前のバージョンでは新しいリリース、特にランタイムに新機能やバグ修正が含まれるリリース時に開発者がテストを行う際は、既存のアプリケーションに影響を与えないよう新しいリリースのインストール前に、新しい環境を構成、もしくは既存環境をバックアップする必要がありました。今回、ランタイムが独立したコンポーネントとして分離されたため、同じマシンに複数バージョンのランタイムをインストールが可能で、開発者は IDE でワンクリックするだけで新しいバージョンのランタイムに切り替えることができ、問題が見つかった場合は数秒で元のランタイムに戻すことができます。またテストが正常に行われた場合、アプリケーション実行可能ファイルとともに新しいランタイムファイルを配布し、アプリケーション実行可能ファイルを再コンパイルせずに、すぐに新しいランタイムを使用できるようにアプリケーション実行可能ファイルを構成できます。

1.1.2 分離による変更

ランタイムと IDE を分離するため、次の変更が行われました：

- ランタイムファイル名 (.dll、.ini、.pbx、.pbd など含む) が変更され、ファイル名に付加されていたバージョン番号インジケーター ("170"、"190" など) が削除されます。たとえば、pbvm190.dll は pbvm.dll に変更され、pbdom190.pbd は pbdom.pbd に変更されます。

  アプリケーションで pbwsclient.pbd、pbdom.pbd、pbsoapclient.pbd、pbejbclient.pbd を使用している場合は、それに応じてファイル名とパスを更新してください。

- ランタイムファイルを IDE ファイルから物理的に分離するため、以前は “Shared” フォルダーにインストールされていたすべてのランタイム DLL が、"Runtime [バージョン]" フォルダーと ”IDE” フォルダーに別々にインストールされるようになりました。

  "Runtime [バージョン]" フォルダーには、実行時に必要なライブラリが含まれています。場所の例：C:¥Program Files (x86)¥Appeon¥Common¥PowerBuilder¥Runtime 19.2.0.2556

  "IDE" フォルダーには、IDE に必要なサポートファイルが含まれています。場所の例：C:¥Program Files (x86)¥Appeon¥PowerBuilder 19.0¥IDE

  外部プログラムがランタイムファイルを呼び出した場合、プログラムが依存する DLL を正常に検出するため、それに応じてコードを変更する必要がある場合があります (コードサンプルの表示)。

- ランタイムファイルの場所は、システム環境変数 PATH の代わりにシステムレジストリに記録されます。

  32-bit OS の 32-bit アプリは、次のレジストリを検索します：HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥PowerBuilder Runtime

  64-bit OS の 32-bit アプリは、次のレジストリを検索します：HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥Sybase¥PowerBuilder Runtime

  64-bit OS の 64-bit アプリは、次のレジストリを検索します：HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥PowerBuilder Runtime

- PowerBuilder IDE (およびアプリケーション実行可能ファイル) は、使用するランタイムファイルのバージョンを選択できるため、開発者は複数プロジェクトを簡単に保守できます。IDE は、システムオプションを使用して複数バージョンのランタイム (マイナーとメジャーの両方) 用にコンパイルできます。詳細な手順については、*Application Techniques* のセクション 9.2.4.3「Selecting a version of PowerBuilder Runtime」を参照してください。

1.1.3 ランタイムのインストールと切り替え

"PowerBuilder ランタイム" は常に %AppeonInstallPath%¥Common¥PowerBuilder¥Runtime [バージョン] にインストールされます。複数バージョンの "PowerBuilder ランタイム" をインストールして、互換性のある PowerBuilder IDE のバージ

ョンを選択できます。デフォルトでは、互換性がある最新の PowerBuilder ランタイムが PowerBuilder IDE によって使用されます。PowerBuilder IDE とアプリケーション実行可能ファイルのランタイムファイルのバージョンを数秒で簡単に切り替えることができます。PowerBuilder ランタイムのインストールについては、*Application Techniques* のセクション 9.2.4.2「Installing PowerBuilder Runtime」を参照してください。

PowerBuilder IDE によってロードされる PowerBuilder ランタイムと、ネイティブ C/S アプリケーションによってロードされる PowerBuilder ランタイムを選択するには、*Application Techniques* 9.2.4.3「Selecting a version of PowerBuilder Runtime」を参照してください。

PowerBuilder IDE は、同じバージョンまたは以前のバージョンの PowerBuilder ランタイムと連携できます。例えば、以下となります：

表 1.1:

IDE (MR 含む)	サポートされるランタイム (MR 含む)	非サポートランタイム
2019 R3	2019 R3	2021 以降
2021	2019 R3, 2021	2022 以降
2022	2019 R3, 2021, 2022	2022 R2 以降
2022 R2	2019 R3, 2021, 2022, 2022R2	2022 R3 以降
2022 R3	2019 R3, 2021, 2022, 2022R2, 2022 R3	2023 以降

1.1.4 関連機能

ランタイムと IDE の分離に伴い、以下の変更が行われました。

- Environment オブジェクトに新しいプロパティ RuntimePath が追加され、現在のアプリケーション実行可能ファイルで使用されるランタイムパスとバージョンを取得できます。
- 現在のコンピューターにインストールされているランタイムのバージョン番号を取得するために、システム関数 GetInstalledRuntimes が追加されています。
- システムオプションに “PowerBuilder ランタイム変更後にターゲットを開いた場合、ターゲットをフル構築するプロンプトを表示” が追加されました -- IDE の PowerBuilder ランタイムのバージョンを変更して IDE を再起動するたびに、アプリケーションをフル構築するように求められ

ます。また IDE でフル構築を実行すると、新しいランタイムバージョンは IDE でのみ使用される内部プロパティ “appruntimeversion” に記録されます (PBC、OrcaScript、または IDE から exe を作成しても ”appruntimeversion” プロパティのバージョン番号は更新されないことに注意してください) 。

- OrcaScript コマンドを使用する場合、*runtime_version* 引数を使用して、アプリケーション実行可能ファイルのコンパイルに使用される PowerBuilder ランタイムのバージョンを指定できます (例 : OrcaScr190 /D runtime_version="19.2.0.2558" C:¥test¥test.bat)。詳細については、*Users Guide* のセクション 8.3.2「OrcaScript Commands」を参照してください。
- PowerBuilder コンパイラ (PBC) の使用時に /rt 引数 (たとえば、/rt 19.2.0.2558) を使用してアプリケーション実行可能ファイルのコンパイルに使用する PowerBuilder ランタイムのバージョンを指定できます。また、PowerBuilder ランタイムのインストール後に PowerBuilder コンパイラをインストールする必要があります。詳細については、*Users Guide* のセクション 8.2「Appendix B. PowerBuilder Compiler」を参照してください。

- PowerBuilder 2019R3 ランタイムパッケージャツールで生成された MSI / MSM ファイルが拡張され、同じメジャーバージョン (2019 R3 GA 以降) の異なるビルドのランタイムファイルを同じコンピューターにインストールして共存できるようになりました。たとえば、2019R3 と 2019R2 は共存可能で、複数の 2019 R3 MR は共存できます。また MSI ファイルは、システム環境変数 PATH にランタイムファイルパスを設定しなくなりました。したがって、ユーザーはアプリケーション実行可能ファイルによってロードされるランタイムファイルのビルドを決定し、アプリケーション実行可能ファイルとランタイムファイルを同じフォルダーに配置する必要があります。

1.1.5 コードサンプル

例 1:

たとえば、PowerBuilder IDE がインストールされている場所からランタイム DLL ファイルである PBORC190.dll をロードする場合、2019R3 と 2019R2 は異なる方法でコードを記述します。

また、2019R2 のファイル名 PBORC190.dll は 2019R3 では PBORC.dll に変更されていることにも注意してください。

2019 R2 の元のコード :

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

2019 R3 で変更されたコード：

このコードサンプルでは、IDE とランタイムのパスをハードコードしています。レジストリからディレクトリを読み取る場合は、次の例を使用してください。

```
//はじめに、AddEnvironmentPath メソッドを定義
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;
}

//ランタイム DLL をロードする前にランタイムファイルのパスを追加
//PB のアップグレード時にパスを更新
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")
```

または、ハードコードされたパスの代わりにレジストリから IDE とランタイムパスを読み取る次のようなコードを 2019R3 で記述することができます:

```
//はじめに、AddEnvironmentPath メソッドを定義
//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;

    //レジストリからランタイムパスを読み取り
    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);

    //レジストリから IDE パスを読み取り
    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;
}
```

つまり、バージョン 2019R3 では 2 つのステップでコードを記述する必要があります：

ステップ 1：環境変数 PATH にパスを追加できるメソッドを定義します。

ステップ 2：メソッドを呼び出して、PowerBuilder ランタイムおよび PowerBuilder IDE ディレクトリを環境変数 PATH に追加してから、DLL ファイルをロードします。

IDE またはランタイムのパスがスクリプトにハードコーディングされている場合、PowerBuilder ランタイムと IDE を新しいバージョンにアップグレードする場合は、それに応じてパスを更新してください。

例 2：

クラス名によって PowerBuilder のバージョンを決定する以下のようなソースコードがある場合は、コードを変更する必要があります。

2019 R2 の元のコード：

```
Environment le_env
GetEnvironment(le_env)
//Microhelp バー内のウィンドウ位置を決定
ii_width = st_time.x + st_time.width + 25
this.width = ii_width
//オブジェクトクラス名をセット
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
//親子関係をセット
this.wf_setparent()
```

2019 R3 で変更されたコード：

```
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 以降：
                           is_classname = "FNHELP"
                  else //pb2019 R2 以前 :
                           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

//Microhelp ハンドルを取得
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 デプロイメント

PowerBuilder 2019 R3 では、新しいプロジェクトタイプである PowerClient が導入されています。PowerClient は、PowerBuilder アプリケーションのデプロイと更新方法を変更します。アプリケーションがクライアント / サーバーまたはクラウドネイティブアーキテクチャであるかどうかに関係なく、PowerBuilder クライアントは HTTP / HTTPS を介して Web サーバーからデプロイされ、自動的に更新されます。これにより、インストールプログラムの作成、ユーザーへのアプリのデプロイ、アプリを最新の状態に維持するための悩みやコストが排除されます。

おもな機能：

- 管理者権限不要のシームレスなインストール
- フレキシブルな更新戦略による自己更新
- すべての必要なファイル (PBVM、OCX、DLL など) のパッケージ化

- 自動ファイル暗号化と整合性検証
- 任意のオペレーティングシステム上の任意の Web サーバーとの互換性
- FTP 経由で Web サーバーにデプロイ、または .ZIP パッケージを作成

Cloud App Launcher：

Cloud App Launcher は、最初にユーザーのマシンにインストールする必要があるランチャープログラムです。これにより、HTTP / HTTPS を介した PowerBuilder アプリケーションの初期インストールとその後の更新が簡単になります。

2 つのランチャータイプがあります。十分にテストして、技術要件に最適なものを確認することをお勧めします：

- バックグラウンドサービスのないランチャー：このランチャープログラムは、バックグラウンドサービスを使用しません。そのため、インストールと利用が簡単で管理者権限は必要ありません。ただし、ブラウザーに一定の依存関係があるため、使用するブラウザーと構成によってインストール時の操作性が異なる場合があります。
- バックグラウンドサービス付きランチャー：ランチャープログラムはバックグラウンドサービスを使用します。クライアントマシンに複数ユーザー存在する場合、ランチャーをインストールするには管理者権限が必要であり、バックグラウンドサービスと連携します。このランチャータイプは、ブラウザーに依存しません。

コンパイルの違い：

PowerClient は、p コードファイルのコンパイルのみをサポートします。コンパイルは、従来の p コードアプローチとは異なります。すべての PBD ファイルは、個々のオブジェクト / 定義ファイルに非常に細分化されています。たとえば、各 SRW、SRD、SRU などのファイルには、モノリシックな PBD ファイルではなく、対応する個別の p コードファイル (.dwo、.apl、.fun、.win、.udo などの新しいファイル拡張子を持つ) があります。また、デフォルトで p コードファイルは暗号化されているため、簡単に逆コンパイルすることはできません。

詳細については、ホワイトペーパー：Automating On-Premise Deployment of PowerBuilder Apps を参照してください。

PowerClient を使用して、次のことができます：

- PowerClient プロジェクトの定義
- PowerClient プロジェクトのビルドとデプロイ
- PowerClient プロジェクトのインストールと実行
- PowerClient プロジェクトのパッケージ
- PowerClient プロジェクトのアンインストール

詳細な手順については、*Users Guide* のセクション 7.3.7「Creating a PowerClient project」を参照してください。

PowerClient プロジェクトを作成する手順については、*Users Guide* のセクション 7.3.7.10「Tutorial: deploying your first PowerClient project」を参照してください。

注

PowerClient 関連の機能を使用するには、Professional または CloudPro のライセンスが必要です。

注

PowerClient 関連の機能を使用するには、管理者として PowerBuilder IDE を実行する必要があります。

1.2.1 関連機能

アプリケーションが p コード / マシンコードを使用してコンパイルされたネイティブ C/S アプリケーションか、PowerClient を使用してデプロイされたアプリケーションかをチェックするために、2 つのシステム関数 IsPBApp と IsPowerClientApp が追加されました。

1.3 ソースコントロールの機能強化

PowerBuilder のソースコントロールは、次の点が改善されています：

- (Git および SVN)「ワークスペースに接続」ダイアログに [ダウンロード後のソースコードですべての PBL を更新する] オプションが追加されました。PBL が最新のソースコードで更新されている

かどうか不明な場合は、ワークスペースからソースコードをダウンロードした後、すべての PBL を更新するように選択できます。

- (Git のみ)TortoiseGit を使用してブランチを作成したり切り替えたりするのと同じように、PowerBuilder IDE でブランチの作成や切り替えが可能です。詳細については、*Users Guide* のセクション 1.3.3.11「Use branches」を参照してください。

- (Git のみ) ブランチの情報は PowerBuilder IDE に表示されます。たとえば、現在のブランチの名前は、[コミット] メニュー、「コミット」ダイアログのタイトル、および出力ウィンドウに表示されます。

- (Git および SVN) プロジェクトをソースコントロールサーバーに再度バインドすることなく、プロジェクトをそのソースコントロールバインディングとともに新しい場所に移動できます。2019 R3 より前のバージョンでは、プロジェクトが新しい場所に移動されるか、PowerBuilder IDE が再インストールされると、既存のソースコントロールバインディングが無効になるので、バインディングを再確立するにはソースコントロールサーバーに接続してプロジェクトをダウンロードする必要があります。2019 R3 以降のバージョンでは、プロジェクト (.svn または .git および ws_objects サブフォルダーを含む) を移動した後、すぐにソースコントロールを続行できます。

  さらに、プロジェクトが TortoiseSVN や TortoiseGit などのサードパーティツールを使用してソース管理下にある場合は、プロジェクトを移動して (.svn または .git サブフォルダーを含む)、ソース管理タイプ (SVN または Git) を選択できます。ワークスペースのプロパティ [ソース管理] タブで、PowerBuilder IDE がソース管理を追加して続行できるようにします。詳細については *Users Guide* の 1.3.3.12 「View/Edit the connection settings」を参照してください。

- (Git および SVN) ワークスペースをソース管理サーバーに追加するときに、"ws_objects" のソースコードファイルのエンコード形式を指定できます。ANSI/DBCS、HEXASCII、および UTF8 から選択できます。

- (Git および SVN) [競合の編集] オプションを使用して、サードパーティのツールを介して競合を編集できるようになりました。最初にサードパーティツールを指定する必要があります。ワークスペースのプロパティダイアログボックスで、[ソース管理] タブを選択し、[高度設定] ボタンをクリックします。「ソースコントロール詳細設定」ダイアログボックスで、左側のパネルの [ログ / 競合の編集を表示] を選択し、TortoiseSVN または TortoiseGit の実行可能プログラムを指定し

ます。詳細については、*Users Guide* のセクション 1.3.3.9 「Tools for Show Log¥Edit Conflicts」を参照してください。

オブジェクトが競合している場合は、オブジェクトをコミットしないでください。また、競合を編集した後、オブジェクトを更新して変更をインポートする必要があります。競合を解決した後にエラーが発生した場合は、プロジェクトの再生成またはフル構築をお勧めします。

- (Git のみ) PowerBuilder IDE は、autocrlf オプションが追加されました。他の Git ツールがそのようなオプションをインストールおよび設定していない場合、Windows では true に設定します。ただし、autocrlf の設定を手動で変更して (true から false または input に)、サーバーからオブジェクトをダウンロードした場合、これらのオブジェクトは CRLF ではなく LF で行が終わり、PowerBuilder IDE でコンパイルエラーが発生します。このようなコンパイルエラーが発生した場合は、autocrlf を true に設定し、サーバーからファイルを再度ダウンロードしてから、再度コンパイルする必要があります。詳細については、*Users Guide* のセクション 1.3.3.4 「Get objects from Git」を参照してください。
- (Git および SVN) プロジェクトディレクトリには、マルチバイト文字を含めることができます。
- (Git および SVN) PowerBuilder IDE でのソース管理中に発生したエラーメッセージは、「出力」ウィンドウの [エラー] タブに一覧表示されます。以前のバージョンでは、エラーは「出力」ウィンドウの [デフォルト] タブにのみ表示されていました。

•

1.4 ライセンスの強化

PowerBuilder のライセンスメカニズムは、次の点が改善されています：

- (オンラインおよびオフラインライセンス) PowerBuilder 2019 R3 以降、同じバージョンの複数のインスタンスにログインするのと同じように、1 つのユーザーアカウントで同じマシン上の複数のバージョンに同時にログインできます。

  以前のバージョンでは、ユーザーアカウントを使用して 1 つのバージョン (2019 R2 とします) にログインしている場合、このアカウントが 2019 R2 からログアウトされない限り、この同じアカウントを使用して同じマシン上の別のバージョン (2017 R3 など) にログインすることはできません。2019 R3 以降のバージョンでは、同じアカウントを使用して、同じマシン上の複数のバージョンに同時にログインできます。たとえば、2019 R3 を正常にインストールしてログインし、2021 をインストールした場合、2019 R3 からログアウトしていなくても、同じユーザーアカウントを使用して 2021 に自動的にログインできます。

- (オンラインライセンスのみ) コマンドラインパラメーターでユーザーアカウントとパスワードを指定することにより、PowerBuilder IDE を起動してログインができるようになりました。コマンドラインモードはオンラインライセンスでのみ機能します。

  コマンドラインパラメーターを使用して、ユーザーアカウントを直接指定できます。例：

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

  /AC -- ユーザーアカウント

  /PW -- パスワード

  /RC -- 資格情報の記憶 (Y または N)。デフォルトは Y です。/ALS が Y の場合、/RC は常に Y です。このため /ALS が Y に設定されている場合は /RC は無視されます。このパラメーターは、/AC と /PW の両方が設定されている場合に有効になります。

  /ALS -- 起動時の自動ログイン (Y または N)。デフォルトは Y です。このパラメーターは、/AC と/PW の両方が設定されている場合にのみ有効になります。

  /SOE -- 終了時にサインアウト (Y または N)。デフォルトは Y です。このパラメーターは、/AC と/PW の両方が設定されている場合にのみ有効になります。

  コマンドラインパラメーターを使用して、ファイル (ユーザーの資格情報を含む) を指定することもできます。例：

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

/LIF -- 暗号化されたパスワードおよびその他のログイン設定を含むライセンスログイン初期化ファイルへのフルパスです。このファイルは、独立したツール (%AppeonInstallPath%¥PowerBuilder [バージョン]¥Tools¥LoginIniFileCreator.exe) によって作成されます。このツールでは、ユーザーアカウント、パスワード、ログイン設定、およびファイルパスを指定できます。ログイン資格情報を保護するために、生成されたファイルでパスワードが暗号化されます。(ツールを他の場所にコピーする場合は、実行可能ファイルと、msvcp100.dll ファイルおよび msvcr100.dll ファイルを同じフォルダーにコピーしてください)。

注

ユーザーアカウントがすでに正常にログインしている (そしてそれ以降ログアウトしていない) 場合、コマンドラインで指定すると、そのユーザー名とパスワードのみが認証されます。コマンドラインで指定された他の設定 (起動時の自動ログイン、終了時のサインアウト、資格情報の記憶など) は無視されます。

- Windows Server オペレーティングシステムでオフラインライセンスをアクティベートすることはできなくなりました。

詳細については、Appeon License User Guide を参照してください。

1.5 デモアプリの変更

PowerBuilder 2019 R3 では次の従来のデモアプリを削除しました：Benchmark、DWGradientTransparency、DWRichTextEditStyle、MDIDockingDemo、Mobilink、および TreeView DataWindow と DatePicker。このバージョンでは次の 3 つのデモアプリのみを提供します。これらは Windows のスタートメニューの Appeon PowerBuilder 2019 R3 からアクセスすることができます。

- **Example App** -- このデモアプリはこれまでのバージョンと同じです。
- **Example Sales App** -- このデモアプリは下記の実装例として 2019 R3 で新たに追加されました。
- RESTClient オブジェクトを使用して RESTful Web API とデータを交換する方法

-
  - PowerBuilder UI Theme 機能を使用して UI のルックアンドフィールを最新化する方法
  - PowerBuilder RibbonBar コントロールを使用して最新の方法でメニューを整理する方法
  - **Example Graph App** -- このデモアプリは 2019 R3 で新たに追加されました。次の機能を使用しています。
    JavaScript クラスとして公開されているサードパーティのビジュアルチャート (Google Charts、Apache EChart など) をレンダリングするための PowerBuilder WebBrowser コントロール。これらの JavaScript / HTML5 チャートは、PowerBuilder グラフデータウィンドウコントロールの有用な補足として使用できます。
    このアプリには、2 つのアプリケーションターゲットが含まれています。1 つは Google Charts 用、もう 1 つは Apache EChart 用で、次の方法を実装したデモです。
  - PowerBuilder WebBrowser コントロールを介して JavaScript を実行し、さまざまなスタイルのグラフを生成してデータ表示を動的に表現する方法
  - JavaScript と PowerScript が相互作用できるように、チャートイベントを WebBrowser コントロールのイベントに接続する方法

    このデモアプリには、Google Charts と Apache EChart (他の JavaScript / HTML5 チャートも同様) を PowerBuilder アプリケーションに組み込む方法を詳細な手順で説明する readme ファイルが含まれています。

1.6 PowerBuilder IDE の機能強化

PowerBuilder IDE は、次の機能が拡張されています：

オブジェクトブラウザでオブジェクト (アプリケーション、データウィンドウオブジェクト、ウィンドウ、メニュー、ユーザオブジェクト、関数など) をダブルクリックすることで、そのオブジェクトに関連付けられているペインタを開くことができます。これまではオブジェクトを右クリックし、[編集] を選択してペインタを開く必要がありました。

- データウィンドウのカラムが「ドロップダウンデータウィンドウ」編集様式に設定され、データウィンドウオブジェクトに関連付けられている場合、カラムを右クリックし、[**DropDownDW を変更...**] を選択することで、関連付けられたデータウィンドウオブジェクトを直接ペインタで開いて編集できます。

- PowerBuilder のシステムダイアログ (アプリケーション、システムオプション、さまざまなオブジェクトのオプションダイアログなど) のレイアウトは再配置されており、ダイアログのサイズをドラッグにより調整できます。

- データウィンドウのヘッダ区域にあるテキストコントロールの Enabled プロパティがデフォルトで有効となり、テキストコントロールのタブ順序はデフォルトで 0 に設定されます。これまでは、Enabled プロパティがデフォルトで有効ではなく、タブの順序は設定されていませんでした。

1.7 RibbonBar の機能強化

PBR (PowerBuilder リソース) ファイルを作成して、RibbonBar コントロールで使用されるリソース名 (BMP、PNG など) を一覧表示し、アプリケーション実行可能ファイルとともに PBR ファイルをコンパイルできます。

1.8 PBU 変換で long 型をサポート

PBU 変換関数 (UnitsToPixels および PixelsToUnits) は、第 1 引数で long 型をサポートします。第 1 引数の値は integer 型または long 型を利用できます。第 1 引数の値が数値の場合、戻り値はデフォルトで long 型です。

1.9 WebBrowser の機能強化

WebBrowser コントロールは、次の関数とイベントによる JavaScript の実行をサポートしています：

- EvaluateJavascriptAsync 関数 – JavaScript を非同期で実行します。この関数は、EvaluateJavascriptFinished イベントをトリガーします。

```
EvaluateJavascriptAsync (string javascript)
```

- EvaluateJavascriptSync 関数 – JavaScript を同期的に実行します。

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

- EvaluateJavascriptFinished イベント – EvaluateJavascriptAsync 関数が実行された後にトリガーされます。

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

-
- RegisterEvent 関数 – JavaScript でトリガーできるように、ユーザー定義のイベントを登録します。

```
RegisterEvent (string eventname)
```

  PowerBuilder のユーザイベントは、1 つの文字列型パラメーターと文字列型の戻り値で定義できます。詳細については、セクション 4.5.3「Defining user events for WebBrowser」を参照してください。

- UnregisterEvent 関数 – RegisterEvent 関数を使用して登録されたユーザ定義イベントの登録を解除します。

```
UnregisterEvent (string eventname)
```

WebBrowser コントロールは、ベーシック認証とダイジェスト認証をサポートしています：

- アクセスする Web ページにベーシック認証またはダイジェスト認証が必要な場合、WebBrowser コントロールは、ユーザーが認証用のユーザー名とパスワードを入力するためのログインウィンドウを自動的に表示します。認証が失敗した場合、認証が成功するか認証操作がキャンセルされるまでこのウィンドウが再度表示されます。

WebBrowserSet 関数と WebBrowserGet 関数は、ローカルファイルとリソースへのアクセスをサポートするように拡張されています：

- allow-file-access-from-files -- ローカルファイル (XML など) へのアクセスを許可するかどうか。

- enable-media-stream -- マイクまたはカメラへのアクセスを許可するかどうか。

1.10 JSONParser および JSONGenerator の機能強化

1. パスによる項目の取得と設定

2. JsonNaN (無効な数)、JsonPositiveInfinity (正の無限大)、JsonNegativeInfinity (負の無限大) などの特別な値の処理

上記をサポートするために、JSONParser オブジェクトは次の関数を追加または改善しました：

- 新しい GetNumberType 関数 -- 数値項目の値型を決定します：JsonNumber、JsonNaN、JsonPositiveInfinity、または JsonNegativeInfinity。

- 次の関数は、JsonNaN、JsonPositiveInfinity、および JsonNegativeInfinity をサポートするように拡張されています：

    - LoadString は JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値を含む文字列の読み込みをサポートします

    - LoadFile は JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値を含むファイルの読み込みをサポートします

    - GetItemObjectJSONString および GetItemArrayJSONString JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値を含むオブジェクトまたは配列の戻り値をサポートします

    - GetItemNumber は JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値の取得をサポートします

    - 新しい GetItemByPath 関数 -- パスに従って項目のハンドルを取得します。

- 新しい ContainsPath 関数 -- パスが存在するかどうかを確認します。

- 新しい GetItemObjectJSONString および GetItemArrayJSONString 関数 -- オブジェクト項目または配列項目の文字列値 (JSON 形式) を取得します。

- 次の関数は、パスで項目を取得するように拡張されています：GetChildCount、GetChildKey、GetChildItem、GetItemArray、GetItemBlob、GetItemBoolean、GetItemDate、GetItemDateTime、GetItemNumber、GetItemObject、GetItemString、GetItemTime、GetItemType および ContainsKey。

JSONGenerator オブジェクトは、次の関数を追加または改善しました：

- 新しい ImportFile および ImportString 関数 – JSON 文字列またはファイルから JSON 項目をインポートします。

-
- 新しい GetItemByPath および GetPathByItem 関数 -- パスごとに項目のハンドルを取得するか、項目のハンドルごとにパスを取得します。

- 次の関数は、JsonNaN、JsonPositiveInfinity、および JsonNegativeInfinity をサポートするように拡張されました：

    - PASSWORDNumber、ImportFile、および ImportString は、JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値を含むパラメーターをサポートします

    - GetJSONString は、JsonNaN / JsonPositiveInfinity / JsonNegativeInfinity の値を含む戻り値をサポートします

- 次の関数はパスごとに項目を追加するように拡張されました：PASSWORDArray、PASSWORDBlob、PASSWORDBoolean、PASSWORDDate、PASSWORDDateTime、PASSWORDNull、PASSWORDNumber、PASSWORDObject、PASSWORDString、および PASSWORDTime。

1.11 HTTPClient の機能強化

HTTPClient オブジェクトは次の新しいプロパティを追加しました：

- IgnoreServerCertificate -- リクエストの送信時に特定のタイプのサーバー証明書エラーを無視します。

- CheckForServerCertRevocation -- リクエストの送信時にサーバー証明書が取り消されているかどうかを確認します。

1.12 64-bit リッチテキストエディターのサポート

PowerBuilder 2019 R3 は、64-bit バージョンの TX Text Control をサポートしています。ユーザーは、64-bit エディターとして (Microsoft RichEdit Control の代わりに)「組み込み TX Text Control ActiveX 28.0」を選択できます。

詳細については、*Application Techniques* の 4.5.1.3「Rich text editors」を参照してください。

1.13 UI アクセシビリティと自動化のサポート

PowerBuilder は、Windows のアクセシビリティと自動化の技術である Microsoft Active Accessibility (MSAA) と Microsoft UI Automation の 2 つをサポートしています。

MSAA は Windows 95 で導入されたレガシー技術で、PowerBuilder の標準コントロールを適切にサポートしますが、データウィンドウなどの PowerBuilder カスタムコントロールには大きな制限があります。PowerBuilder 2019 R3 から、PowerBuilder は MSAA の制限を乗り越えた、より新しい高性能なテクノロジーをサポートしています。Microsoft UI Automation と呼ばれるこの新しいテクノロジーは、標準コントロールだけでなく、カスタムコントロール (PowerBuilder データウィンドウやデータウィンドウの子コントロールなど) も操作するための豊富なプロパティセットと拡張インターフェイスを提供します。

Microsoft UI Automation テクノロジーを使用すると Windows 10 や Windows Server 2019 において、PowerBuilder コントロールの UI 要素にスクリーンリーダー (Windows ナレーターなど)、アクセシビリティツール (Accessibility Insights、Inspect など) および自動テストツール (QTP など) からアクセスできます。

Microsoft UI Automation で PowerBuilder コントロール / オブジェクトのサポートを有効にするには、コントロールの [プロパティ] タブページでアクセシブル名プロパティとアクセシブル詳細プロパティを設定する必要があります。

Microsoft UI Automation テクノロジーを使用すると、PowerBuilder 標準コントロール (表 1 参照) だけでなく、PowerBuilder データウィンドウコントロール (表 2 および 3 参照) にもアクセスしてテストすることができます。

表 1：Microsoft UI Automation で**サポートされる** PowerBuilder コントロール/オブジェクト

表 1.2：

•

アニメーション チェックボックス コマンドボタン データウィンドウ 日付ピッカー ドロップダウンリストボックス ドロップダウンピクチャリスト エディットマスク	グループボックス 水平プログレスバー 水平スクロールバー 水平トラックバー リストボックス リストビュー 月表示カレンダー マルチラインエディット	ピクチャ ピクチャボタン ピクチャハイパーリンク ピクチャリストボックス ラジオボタン シングルラインエディット スタティックテキスト	スタティックハイパーテキスト タブ ツリービュー ユーザオブジェクト 垂直プログレスバー 垂直スクロールバー 垂直トラックバー

次の PowerBuilder コントロール / オブジェクトは、Microsoft UI Automation では**サポートされていません**：グラフ、エディット、インクピクチャ、線、OLE、楕円、長方形、リボンバー, 丸長方形、リッチテキストエディット、WebBrowser。

表 2：Microsoft UI Automation で**サポートされる**データウィンドウの表示形式

表 1.3：

クロスタブ フリーフォーム グリッド グループ	ラベル 段組み タブラ ツリービュー

次の表示形式のデータウィンドウコントロールは**サポートされていません**：グラフ、OLE 2.0、コンポジット、およびリッチテキスト。

表 3：Microsoft UI Automation で**サポートされる**データウィンドウのコントロール

表 1.4：

ボタン カラム (および以下の編集形式に対応しています。チェックボックス、ドロップダウンリストボックス、ドロップダウンデータウィンドウ、エディットボックス、エディットマスク、ラジオボタン)	計算フィールド グループボックス テキスト

データウィンドウの次のコントロールは、Microsoft UI Automation では**サポートされていません**：グラフ、インクピクチャ、直線、OLE オブジェクト、楕円、長方形、丸長方形、ピクチャ、レポート、および TableBlob。

例

次のステートメントは、ウィンドウ上のコマンドボタンにアクセシブル名およびアクセシブルの詳細プロパティを設定します：

```
cb_1.accessiblename = "削除"
cb_1.accessibledescription = "選択したテキストを削除します"
```

次のステートメントは、データウィンドウオブジェクト上のボタンのアクセシブル名プロパティを設定します：

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

デプロイメント

アクセシブルなアプリケーションをデプロイする際には、pbacc.dll および PBAccessibility.dll ファイルを配布する必要があります。

1.14 PostgreSQL の機能強化

DB ペインタの PostgreSQL データベース関連タスクに次の機能拡張が行われました。

- コンテキストメニューの「システムプロシージャの表示」を使用して、オブジェクトビューで PostgreSQL のシステムプロシージャを表示または非表示にできるようになりました。

1.15 Microsoft SQL Server の機能強化

Microsoft SQL Server データベース関連の機能に次の機能拡張が行われました：

- SQL Server 用の MicrosoftOLE DB ドライバー (MSOLEDBSQL) がサポートされています。データベースプロファイルから **MSOLEDBSQL SQL Server** インターフェイスを選択して、Microsoft OLE DB Driver for SQL Server を使用することで Microsoft SQL Server 2012、2014、2016、2017、または 2019 のデータベースに接続することができます。**MSOLEDBSQL SQL Server** インターフェイスは、SNC SQL Native Client インターフェイスとまったく同じ機能をサポートしています。追加の Microsoft OLE DB Driver の新機能 (SQL Native Client との比較) は、次の PowerBuilder バージョンでサポートされる予定です。

  Microsoft OLE DB Driver for SQL Server 18.2 以降用の Microsoft OLE DB ドライバーがサポートされています。詳細については、*Connecting to Your Database* のセクション 3.4「Using Microsoft SQL Server」を参照してください。

- PowerBuilder アプリケーションは、TLS 1.2 を介して (MSOLEDBSQL SQL SQLServer または SNC SQL Native Client インターフェイスを使用して) Microsoft SQL Server データベースに接続できます。

1.16 インストーラーの機能強化

PowerBuilder インストーラーには、次の変更または機能拡張があります：

- PowerBuilder デモデータベースの実行に (SQL Anywhere エンジンに加えて) PostgreSQL エンジンを使用することを選択できます。

  PostgreSQL エンジンを指定する場合は、PostgreSQL クライアント (PostgreSQL Windows installer のコマンドラインツールに含まれている) と PostgreSQL ODBC ドライバー (32-bit) の両方がインストールされていることを確認してください。

- PowerBuilder ランタイムは、PowerBuilder IDE の個別のコンポーネントとしてインストールできます。PowerBuilder ランタイムと PowerBuilder IDE を 1 つずつ順番にインストールする場合は、PowerBuilder ランタイムを PowerBuilder IDE の前にインストールする必要があります。詳細については、*Application Techniques* のセクション 9.2.4.2「Installing PowerBuilder Runtime」を参照してください。

- PowerBuilder コンパイラインストールプログラムが PowerBuilder インストーラーに統合されたので、PowerBuilder インストーラーから PowerBuilder コンパイラをインストールできます。詳細については、*Users Guide* のセクション 8.2.1「Installing PowerBuilder Compiler」を参照してください。
- 特定のバージョンのすべての製品をフルインストールしていない場合は、インストールを変更できるようになりました。インストーラーのエントリーページの [インストール] ボタンが [変更] に変わります。

1.17 アプリのアップグレードに関する注意事項

このセクションでは、アプリケーションのアップグレードに関するいくつかの重要な注意事項のみを取り上げます。詳細については、Upgrading PowerBuilder Applications を参照してください。

Beta 版から GA にアップグレードする場合

Beta1 または Beta2 から GA にアップグレードする場合は、IDE とランタイムの両方をアップグレードし、GA からインストールされたランタイムを使用してください。GA では「ThirdPartyLegal¥MSRedist」のサードパーティ DLL ファイルを IDE からランタイムに移動しました。Beta 版からインストールされたランタイムを引き続き使用する場合は、ベータ版からインストールされたランタイムを DLL ファイルを ThirdPartyLegal¥MSRedist¥x86 または ThirdPartyLegal¥MSRedist¥x64 からランタイムフォルダー %AppeonInstallPath%¥Common¥PowerBuilder¥Runtime [バージョン] または %AppeonInstallPath%¥Common¥PowerBuilder¥Runtime [バージョン]¥x64 に手動でコピーする必要があります。そうでない場合、GA からインストールされた IDE は、Beta 版からインストールされたランタイムでは機能しません。

Beta1 または Beta2 から GA にアップグレードする場合は、1) サーバーから **PowerClient App Publisher** をアンインストールし、**PowerClient Launcher** (または **Cloud App Launcher)**、**PowerClient Service**、**PowerClient App Shell** などのランチャーとサービスをクライアントからアンインストールします (GA でこれらのプログラムの名前を変更したため)。2) サーバーとクライアントからアプリをアンデプロイ / アンインストールします。 3) GA がプロジェクト設定を変更したため、GA で新しい PowerClient プロジェクトを構成して展開します。Beta 版でデプロイされたアプリは、GA バージョンでは動作しなくなります。

PowerBuilder プロジェクトをアップグレードする場合

PowerBuilder プロジェクトを 2019 / 2019 R2 から 2019 R3 にアップグレードするには、https:// docs.appeon.com/pb2019r3/upgrading_pb_apps/ch04.html を参照してください。

1.18 廃止された機能

廃止された機能は引き続き使用できますが、テクニカルサポートの対象ではなくなり、拡張されなくなります。

これらの廃止された機能の**移行パス**については、Product Availability Matrix ページの廃止 / 廃止された機能を参照してください。

次の機能は、バージョン 2019 R3 以降、廃止された機能と見なされます：

- 組み込みリッチエディットコントロール (TE エディットコントロール)
- Inet オブジェクト
- OLE Control で Internet Explorer ブラウザーページを開く (GetURL、PostURL、HyperlinkToURL)
- Windows 7 オペレーティングシステム

次の機能は、バージョン 2019 R2 以降、廃止された機能と見なされます：

- ドッキング ウィンドウ
- Windows Server 2008 オペレーティングシステム

次の機能は、バージョン 2019 以降、廃止された機能と見なされます：

- .NET Web サービスターゲット (SOAP を使用)
- .NET アセンブリターゲット

次の機能は、バージョン 2017 R3 以降の廃止された機能と見なされます：

- SOAP サーバーに接続するための Web サービスプロキシ (SoapConnection クラス、SoapException クラス、SoapPBCookie クラス、UDDIProxy クラスを含む)
- OData サービスと SOAP Web サービスを使用したデータウィンドウデータソース
- Web データウィンドウ
- EJB クライアント

2 PowerBuilder 2019 R2 の新機能

この章について

この章では、PowerBuilder 2019 R2 の新機能を紹介します。

2.1 更新されたシステム要件

PowerBuilder 2019 R2 は、次のシステムをサポートしています：

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

2.2 新しい RibbonBar コントロール

RibbonBar コントロールを使用すると、ユーザーインターフェイスでユーザーコマンドを整理する最新の方法であるリボンを作成できます。これは、Category、Panel、Group、LargeButton (RibbonMenu の有無にかかわらず)、SmallButton (RibbonMenu の有無にかかわらず)、CheckBox、ComboBox、TabButton (RibbonMenu の有無にかかわらず)、および Spin (現在サポートされていない) で構成されます。開発者が効率的にリボンを作成できるように、「RibbonBar Builder」と呼ばれるツールが提供されています。RibbonBar Builder を使用すると、開発者は、グラフィカル UI をその場でプレビューしながら、XML エディターで RibbonBar コントロールテンプレートを変更できます。

詳細については、*Objects and Controls* のセクション 2.90「RibbonBar control」および *Users Guide* のセクション 4.4「Working with RibbonBar」を参照してください。

2.3 .NET アセンブリの呼び出し

PowerBuilder アプリケーションは、DotNetAssembly と DotNetObject の 2 つのオブジェクトを介して、.NET アセンブリを直接呼び出すことができます。DotNetAssembly は .NET アセンブリをロードするために使用され、DotNetObject はアセンブリ内の .NET クラスにマップする

ために使用されます。PowerBuilder で DotNetAssembly インスタンスと DotNetObject インスタンスを作成した後、.NET クラスで定義された関数を直接呼び出すことができます。
開発者がより生産的な方法でスクリプトを作成できるように、「.NET DLL Importer」と呼ばれるツールが提供されており、開発者が .NET クラスの関数を正しく呼び出すためのスクリプトを作成できます。.NET DLL Importer は、.NET クラス、関数、プロパティ、およびパラメーターの名前とデータ型を .NET アセンブリからアプリケーション PBL にインポートできます。DotNetObject オブジェクトを各 .NET クラスの NVO として作成し、.NET 関数を NVO にインポートします。その後、開発者は NVO を呼び出すスクリプトを記述し、対応する .NET コードを直接実行するように機能することができます。

詳細については、*Application Techniques* のセクション 5.1「Calling .NET Assembly in an Application」、*Objects and Controls* のセクション 2.16「DotNetAssembly object」、および *Objects and Controls* のセクション 2.17「DotNetObject object」を参照してください。

2.4 強化された UI テーマ

個々のコントロール、オブジェクト、ユーザオブジェクト、またはウィンドウに独自のテーマ設定を設定できるようになりました。たとえば、1 つのボタンに他のボタンとは異なるテーマ設定を設定できます。また、同じウィンドウ内の同じタイプのコントロールに独自のテーマ設定を設定できます。たとえば、一方のウィンドウのすべてのグループボックスに、もう一方のウィンドウのグループボックスとは異なるテーマ設定を設定できます。コントロールまたはオブジェクトのテーマ設定を構成する方法については、*Users Guide* の「Custom themes」というセクションを参照してください。

さらに、テーマファイルで次の UI 要素を設定できます：

- カスタムビジュアルユーザオブジェクトの UI 設定
- ウィンドウとユーザオブジェクトの背景色、タイトルバー、境界線、およびシステムボタン (最大化、最小化、復元ボタンなど)
- OLE コントロール、ユーザオブジェクト、およびウィンドウのスクロールバー
- ウィンドウとユーザオブジェクトのメニュー、ツールバー、ステータスバー

2.5 新しい WebBrowser コントロール

WebBrowser コントロールを使用して、次のような Web ブラウザーを作成できます：

- JavaScript を含む Web ページの閲覧をサポート
- HTML および HTML5 ページの閲覧をサポート
- Web ページの一般的な形式でのビデオの閲覧をサポート
- Flash の再生をサポート (ユーザーがフラッシュプラグインをインストールした場合)
- Web ページを PDF として印刷し、印刷イベントに応答することをサポート
- 複数の言語をサポートする Web ページの閲覧をサポート
- さまざまな HTTPS プロトコルをサポート
- 右マウスボタンのコンテキストメニューをサポート
- PDF ファイルのオンライン閲覧をサポート
- Web ページのズームイン / ズームアウトをサポート
- ローカルファイル (htm、gif、jpg、jpeg、png、swf、txt、c、cpp、pdf などを含む) の参照をサポート
- サーバー証明書からのエラーイベントへの応答をサポート
- ファイルのダウンロード場所とプロキシ設定の動的構成をサポート

WebBrowser オブジェクトのサポート、非サポート機能およびプロパティ / 関数 / イベントの詳細については、*Objects and Controls* のセクション 2.150「WebBrowser control」を参照してください。

2.6 強化された CoderObject および CrypterObject

CoderObject オブジェクトに次の新しい関数が追加されました：

- Base32Decode -- Base32 デコーダーを使用して文字列値をデコードします。

- Base32Encode -- Base32 エンコーダーを使用して blob 値をエンコードします。
- Base64UrlDecode -- Base64Url デコーダーを使用して文字列値をデコードします。
- Base64UrlEncode -- Base64Url エンコーダーを使用して blob 値をエンコードします。

CrypterObject オブジェクトに次の新しい関数が追加されました：

- SymmetricGenerateKey -- 非対称アルゴリズムの秘密鍵を生成します。

詳細については、*Objects and Controls* のセクション 2.7「CoderObject object」および *Objects and Controls* のセクション 2.15「CrypterObject object」を参照してください。

2.7 強化された ExtractorObject

ExtractorObject オブジェクトに次の新しい関数が追加されました：

- GetFilesCount -- アーカイブに含まれるファイルの数を取得します。
- GetFilesList -- 圧縮パッケージ内のファイルのリストを取得します。
- Extract -- 指定されたパッケージ内のファイルのみを抽出するか、圧縮パッケージ内の指定されたファイルをバッファーに抽出します。

詳細については、*Objects and Controls* のセクション 2.33「ExtractorObject object」を参照してください。

2.8 強化されたソースコントロール

ソースコントロールには次の拡張機能があります：

(SVN および Git) SVN/Git のコミット (SVN Get/Release Lock、リバート、またはリゾルブ) ダイアログからの差分を表示することをサポートしています。コミット (SVN Get/Release Lock、リバート、またはリゾルブ) ダイアログ -- コミット (SVN Get/Release Lock、リバート、またはリゾルブ) ダイアログで、オブジェクトリストのアイテムをダブルクリックすると、ローカルオブジェクトとソースコントロールと最後に同期したバージョンが比較され、差分が表示されます。

- (SVN および Git) ソース管理でアプリケーションプロパティをサポート -- アプリケーションプロパティ (テーマ設定、リッチテキスト設定など) は、SVN/Git ソース管理で操作できます。

- (SVN のみ) svn:needs-lock プロパティをサポートします -- PowerBuilder IDE は、このプロパティを設定するための直接オプションを提供しません。このプロパティは、SVN クライアント (TortoiseSVN など) を使用して設定でき、その後、PowerBuilder IDE で svn:needs-lock プロパティを持つオブジェクトを操作 (ロック、コミットなど) できます。

- (SVN のみ) ソースコントロールサーバーに接続、アップロード、またはダウンロードするときに、リポジトリ URL、ワークスペースファイル、ユーザー ID での多言語の利用をサポートします。
- (SVN のみ) Get Lock を拡張して、新しいバージョンのオブジェクトがソースコードサーバーに存在する場合に、ユーザーがオブジェクトをロックする前に最初にオブジェクトを更新できるようにします。

- (Git のみ) ブランチの作成と切り替えをサポートします -- PowerBuilder IDE は、ブランチを作成または切り替えるための直接オプションを提供しません。ただし、サードパーティツール (TortoiseGit など) を使用してブランチを作成して切り替えてから、PowerBuilder IDE を再起動してブランチに自動的に切り替えることができます。その後、PowerBuilder IDE のブランチの下にあるオブジェクトを操作できます。詳細な手順については、*Users Guide* のセクション 1.3.3.11「Use branches」を参照してください。

2.9 デフォルトとして NativePDF を使用する

新しいデータウィンドウの場合、NativePDF! (PDFLib を使用) メソッドは、データウィンドウペインタの [ファイル] メニューから [名前を付けて行を保存] を選択してファイルタイプとして「PDF」を選択した場合、または PDF で SaveAs メソッドを使用した場合、ファイルタイプとしてデフォルトで自動的に選択されます。

ただし、GhostScript を使った Distill! 方式をデフォルトで使うようにしている既存のデータウィンドウについて、PDFlib を使って PDF に保存したい場合は、アプリケーションのプロパティダイアログボックスで「いつも NativePDF! 方法 PDF」を選択することができます。このオプションは、 アプリケーション内のすべてのデータウィンドウに対して有効です。詳しくは、*Users Guide* のセクション 6.2.3.1.1「Saving as PDF using NativePDF! method with PDFlib」を参照してください。

2 つの INI 設定 NativePDF_IncludeCustomFont と NativePDF_Valid は無効になります。

さらに、アプリケーション名、作成者、件名、キーワードなど、生成された PDF ファイルのいくつかのプロパティを設定できます。

2.10 AscentialTest を使用したデータウィンドウのテスト

AscentialTest 9.5.2 以降を使用すると、PowerBuilder データウィンドウオブジェクトのテストを自動化できます。 PowerBuilder データウィンドウオブジェクトの自動テストの記述方法について説明している AscentialTest が提供する このページ にアクセスしてください。

2.11 PowerBuilder IDE の機能強化

PowerBuilder IDE に関連するその他の新機能または変更された機能は次のとおりです：

- 新しい画像選択ダイアログ -- 以前は、画像はドロップダウンリストから選択されていました。このバージョンからは、ダイアログボックスから画像が選択されます。ダイアログボックスで、カスタム画像または組み込み画像から選択できます。組み込みの画像はカテゴリ別にグループ化されており、サイズでフィルタリングしたり、名前で検索したりできます。

  名前の末尾に番号が付与されている一部の画像の名前が変更されました (例：ArrangeTables5!)。ArrangeTables2! に名前が変更されましたが、ArrangeTables5! は保持されるため、アプリケーションで参照されている場合でも、以前と同様にプレビューと実行時に表示されますが、ダイアログでは ArrangeTables2 のみを選択できます (ArrangeTables5! を選択できなくなります）。

- オブジェクトを検索するためのオブジェクトブラウザの拡張 -- ユーザーがオブジェクト名を入力し、各タブに一致するオブジェクトのみを表示できるようにするフィルタボックスがブラウザウィンドウに追加されました。検索結果の合計がウィンドウの右下隅に表示されます。

- ワークスペース、ターゲット、およびライブラリの [内包するフォルダーのオープン] メニューの追加 -- [内包するフォルダーのオープン] メニューは、システムツリーまたはライブラリペインタでワークスペース、ターゲット、またはライブラリを右クリックすると使用できます。

- **C# Model Generator** は、独立したプラグインとして PowerBuilder IDE から移動されました。「DataWindow Converter」という名前に変更され、SnapDevelop インストールプログラムからプラグインとしてインストールされます。

- C# プロジェクトを PowerBuilder IDE にロードできなくなりました。SnapDevelop、Visual Studio などの C# エディターで C# プロジェクトを開くことができます。

2.12 PowerBuilder ランタイムの強化

すべての PowerBuilder ランタイム DLL がデジタル署名されました。

2.13 その他の新機能・変更点

PowerScript や PowerBuilder オブジェクトに関連するその他の新機能は以下の通りです：

- シングルラインエディットコントロールでプレースホルダー プロパティをサポート -- プレースホルダープロパティは、入力フィールドに期待される値の短い説明を指定します。プレースホルダーの値は、テキストラベルやヒントとして使用することができるので、UI コントロールの数を減らすことができます。プレースホルダーの値は、1) テキストプロパティが設定されている場合 (テキストの値が最初に表示される)、2) 入力フィールドにフォーカスがある場合には表示されません。

- データウィンドウのチェックボックスとラジオボタンの編集スタイルで、実行時にボックス / 円とテキストの垂直方向のセンタリングが強化されました (デザイン時にはボックス / 円は上部に配置されています)。

- 画像ファイルにファイル拡張子がない場合に画像ファイルを検証および表示するように、ビットマップデータウィンドウ式機能が拡張されました。

- TX Text Control を使用するリッチテキストデータウィンドウでは、データウィンドウの SaveAs 関数を使用して PDF を直接生成できるようになりました。例えば、dw_1.SaveAs ("c:¥dw_one.pdf", PDF!, false) のようになります。

PowerBuilder インストーラーに関連するその他の新規または変更された機能：

- 2019 R2 からは、ユーザーが MR バージョンを選択して、製品版と選択した MR の両方のインストールをワンステップで完了させることができます。

3 PowerBuilder 2019 の新機能

この章について

この章では、PowerBuilder 2019 の新機能を紹介します。

3.1 C# 機能で PowerBuilder を強化する

.NET パッケージ

PowerBuilder DataStore の .NET 切り替えを実装するために、2 つの .NET パッケージ（**PowerBuilder.Data** および **PowerBuilder.Data.AspNetCore**）が提供されています。**PowerBuilder.Data** は、.NET Core と互換性のあるピュアな .NET DataStore と、REST インターフェイスを介して PowerBuilder データウィンドウ または DataStore と統合するための関連ライブラリを提供します。.NET DataStore は PowerBuilder DataStore と同様に動作し、API の命名規則も同じなので、既存のプロジェクト資産を簡単に移植することができます。**PowerBuilder.Data** は、.NET データモデルに対して動作する ModelStore オブジェクトも提供しており、.NET DataStore の代わりに使用することができます。

また、新しい .NET ORM フレームワーク (**SnapObjects** と呼ばれる) を使用して .NET プロジェクトを拡張できます。このフレームワークは、 **PowerBuilder.Data** や **PowerBuilder.Data.AspNetCore** と同様に .NET Core とも互換性があります。

PowerBuilder .NET API および SnapObjects .NET API の詳細については、Documentation Center を参照してください。

C# マイグレーション

.NET DataStore と ModelStore の C# データオブジェクトとモデルを生成するために、**C# Model Generator** と呼ばれるバッチ式の DataWindow / DataStore 変換ユーティリティが提供されています。

また、**PowerBuilder.Data.AspNetCore** の SqlExecutorExtention オブジェクトも提供されており、これを使用して埋め込み SQL を PowerScript プロジェクトから新しい C# プロジェクトに仮想的にコピー＆ペーストすることができます。

C# IDE

比較的フル機能の C# IDE がスタンドアロンで提供されており (**SnapDevelop** と呼ばれる)、PowerBuilder IDE から起動することができます。SnapDevelop は、C# Web API、C# ノンビジュアルアセンブリ、ユニットテスト (xUnit を使用) などのノンビジュアルプロジェクトの開発をサポートしています。また、プロジェクトウィザード、高度なオートスクリプト、C# 言語サービスなど、開発者の生産性を高める強力なツールを提供しています。

SnapDevelop の詳細については、Documentation Center を参照してください。

3.2 新しい UI テーマ

新しい UI テーマシステムは、アプリケーションの UI のレンダリング方法をコードレスで実現するために提供されます。新しい UI テーマの使い方については、*Users Guide* のセクション 2.1.4.5, 「Specifying a theme for the application UI」を参照してください。

システムテーマとカスタムテーマ

4 つの新しいシステムテーマ (Flat Design Blue、Flat Design Dark、Flat Design Grey、Flat Design Silver) が提供され、アプリケーションのウィンドウ、データウィンドウ、すべてのビジュアルコントロール (直線、楕円、長方形、丸長方形、ピクチャ、ピクチャ ハイパーリンク、アニメーションを除く) に適用することができます。また、これらのシステムテーマをカスタマイズしたり、これらのシステムテーマに基づいて独自のテーマを作成することもできます。

テーマの適用

アプリケーションにテーマを適用するには、アプリケーションオブジェクトの [付加的なプロパティ] にある [テーマ] タブで設定するか、ApplyTheme 関数を使用してテーマを動的に設定することができます。

テーマはランタイムで動作し、デザインタイムでは何の効果もありません。

テーマの設定を変更

テーマを適用した後で、特定のコントロールや状態の表示をさらに調整したい場合は、デフォルトまたは指定したディレクトリにあるテーマの「theme.json」ファイルを開き、対応するテーマの設定を変更することができます (注意が必要です)。

システムテーマの設定を元に戻したい場合は、アプリケーションオブジェクトの [付加的なプロパティ] の [テーマ] タブにある [リストア] ボタンを使って行うことができます。[リストア] ボタンは、システムテーマがデフォルトのディレクトリにある場合のみ有効です。

テーマで設定できること

技術的な問題で、テーマを適用しても、いくつかのコントロールやその状態が望ましい UI 効果にならないことがあります。テーマで設定できるものとできないものについては、*Users Guide* のセクション 2.1.4.5「Specifying a theme for the application UI」を参照してください。

3.3 TX Text Control を内蔵エディターとして組み込み

PowerBuilder 2019 からは、特別な OEM 版の TX Text Control ActiveX が PowerBuilder の内蔵リッチテキストエディターとして組み込まれています。これは、PowerBuilder のすべてのエディションで追加費用なしで提供され、 SAP PowerBuilder バージョン 12.6 またはそれ以前の RichTextEdit をすでに利用しているすべての既存の PowerBuilder プロジェクトで、後方互換性の理由から使用することが強く推奨されています。OEM 版 TX Text Control の使用を選択するには、アプリケーションのプロパティダイアログボックスで、[RichTextEdit] タブを選択し、最初のオプション「組み込み TX Text Control」を選択します。

3.4 新規 / 強化された PowerBuilder オブジェクト

3.4.1 RESTClient オブジェクトの強化

RESTClient オブジェクトに以下の機能が追加されました：

•Submit は、アプリケーションクライアントから RESTful Web サービスにデータを送信できるだけでなく、RESTful Web サービスからレスポンスボディを取得することもできます。

送信するデータは、データウィンドウ、データストア、データウィンドウチャイルド、または JSONPackage のいずれかです。データを送信するデータウィンドウバッファは、1 つまたは複数指定することができ、データウィンドウの範囲も指定できます。このリクエストは、OAuth 2.0 認証をサポートしています。

```
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：HTTP DELETE リクエストをサーバーに送信し、サーバーからのレスポンスの内容を取得します。

  これまでは、HTTPClient オブジェクトのみが RESTful Web サービスへのリクエスト送信をサポートしていました。今回、RESTClient オブジェクトから直接リクエストを送信できるようになり、リクエストは OAuth 2.0 の認証をサポートしています。

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

- SendGetRequest：HTTP GET リクエストをサーバーに送信し、サーバーからのレスポンスの内容を取得します。

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

- SendPatchRequest：HTTP PATCH リクエストをサーバーに送信し、サーバーからのレスポンスの内容を取得します。

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

- SendPostRequest：HTTP POST リクエストをサーバーに送信し、サーバーからのレスポンスの内容を取得します。

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

- SendPutRequest：HTTP PUT リクエストをサーバーに送信し、サーバーからのレスポンスの内容を取得します。

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

- GetJWTToken：POST メソッドで JWT トークンを取得します。

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

- SetJWTToken：サーバーに送信される HTTP リクエストヘッダーに JWT トークン文字列を設定します。

```
SetJWTToken(string jwtToken)
```

- GetOAuthToken：OAuth 2.0 アクセストークンを取得します。

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

- SetOAuthToken：サーバーに送信される HTTP リクエストヘッダに OAuth 2.0 のトークン文字列を設定します。

```
SetOAuthToken(string token)
```

- RetrieveOne：RESTFul Web サービスからデータウィンドウ、データウィンドウチャイルド、またはデータストアに 1 つのデータ行を取得します。

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

RESTClient オブジェクトの以下の機能が変更されました：

- Retrieve - RESTFul Web サービスからデータウィンドウ、データウィンドウチャイルド、またはデータストアにデータを取得します。

  RESTful Web サービスから受信したデータが gzip として圧縮されている場合、自動的に展開されます。現在、gzip 圧縮形式のみがサポートされています。SetRequestHeader 関数を使用して、gzip 圧縮形式のみを許可するように Accept-Encoding ヘッダーを設定できます。

- SetRequestHeader - リクエストヘッダーの追加、または既存のリクエストヘッダーが存在する場合はその値の追加 / 置換をサポートします。

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

これらの機能の詳細については、*Objects and Controls* のセクション 2.87「RESTClient object」を参照してください。

3.4.2 HTTPClient オブジェクトの強化

HTTPClient オブジェクトの以下の機能が強化されました：

- SetRequestHeader：リクエストヘッダーの追加、または既存のリクエストヘッダーが存在する場合はその値の追加 / 置換をサポートします。

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

- SendRequest：ユーザーが Content-Type リクエストヘッダーで指定した charset でデータをエンコードすることをサポートします。charset が指定されていない場合、この関数はデフォルトで UTF-8 でデータをエンコードします。

- GetResponseBody：Content-Type リクエストヘッダーでユーザーが指定した文字セットを使用してデータをエンコードすることをサポートします。charset が指定されていない場合、こ

の関数は BOM ヘッダーに基づいてエンコードタイプを判別し、データを UNICODE に変換します。

これらの機能の詳細については、*Objects and Controls* のセクション 2.41「HTTPClient object」を参照してください。

3.4.3 JSONPackage オブジェクトの強化

JSONPackage オブジェクトの GetValue 関数は常に結果を文字列で返し、SetValue 関数は文字列型の値のみを設定します。JSONPackage オブジェクトには、さまざまな種類の値を取得または設定するための関数が追加されました。

- GetValueBlob – blob のキー値を取得します。
- GetValueBoolean - boolean のキー値を取得します。
- GetValueDate - date のキー値を取得します。
- GetValueDateTime - datetime のキー値を取得します。
- GetValueNumber - number のキー値を取得します。
- GetValueString - string のキー値を取得します。
- GetValueTime - time のキー値を取得します。
- SetValueBlob - blob のキー値を設定します。
- SetValueBoolean - boolean のキー値を設定します。
- SetValueDate - date のキー値を設定します。
- SetValueDateTime - datetime のキー値を設定します。
- SetValueNumber - number のキー値を設定します。
- SetValueString - string のキー値を設定します。
- SetValueTime - time のキー値を設定します。

JSONPackage オブジェクトからデータウィンドウに直接値を取り込んだり、データウィンドウから JSONPackage オブジェクトにキー値を設定したりすることができるようになりました。

- GetValueToDataWindow - キー値を取得し、データウィンドウコントロール、データストアオブジェクト、またはデータウィンドウチャイルドオブジェクトに挿入します。
- SetValueByDataWindow - データウィンドウコントロール、データストアオブジェクト、またはデータウィンドウチャイルドオブジェクトのデータを使用してキー値を設定します。

JSONPackage オブジェクトは、JSONParser オブジェクトの GetItemType 関数と同じように機能する GetItemType 関数も提供します。

- GetItemType - アイテムの種類を取得します。

JSONPackage (および JSONGenerator) の SaveToFile 関数と GetJsonBlob 関数は、結果の blob 文字エンコードを指定できるように拡張されています。

- SaveToFile

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

- GetJsonBlob

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

新しく追加または拡張された関数の詳細については、*Objects and Controls* のセクション 2.47「JSONPackage object」を参照してください。

次の新しいプロパティが JSONPackage オブジェクトに追加されました (JSONParser にも追加されました)：

- ReturnsNullWhenError - エラーが発生したときに値取得関数が NULL 値を返すかどうかを指定します。

  このプロパティを使用すると、getitem 関数が null を返した場合に例外がスローされないようにすることができます。

  詳細については、*Objects and Controls* のセクション 3.237「ReturnsNullWhenError」を参照してください。

3.4.4 JSONGenerator の強化

JSONGenerator (および JSONPackage) の SaveToFile 関数と GetJsonBlob 関数は、結果の blob の文字エンコードを指定できるように拡張されています。

- SaveToFile

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

- GetJsonBlob

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

詳細については、*PowerScript Reference* のセクション 2.4.664「SaveToFile」および *PowerScript Reference* のセクション 2.4.286「GetJsonBlob」を参照してください。

3.4.5 JSONParser オブジェクトの強化

次の新しい関数が JSONParser オブジェクトに追加されます：

- ContainsKey - キー名が存在するかどうかを確認します。

  この関数を使用して、GetItem などの他の関数を実行する前に、JSONParser オブジェクトに特定のキーが存在するかどうかを確認できます。

  詳細については、*PowerScript Reference* のセクション 2.4.91「ContainsKey」を参照してください。

次の機能が強化されています：

- GetItemType - アイテムの種類を取得します。

  子アイテムのキーを指定して、子アイテムの種類を直接取得できるようになりました。

  詳細については、*PowerScript Reference* のセクション 2.4.285「GetItemType」を参照してください。

次の新しいプロパティが JSONParser オブジェクトに追加されます (JSONPackage にも追加されます)：

- ReturnsNullWhenError - エラーが発生したときにアイテム取得関数が null 値を返すかどうかを指定します。

このプロパティを使用すると、getitem 関数が null を返した場合に例外がスローされないようにすることができます。

詳細については、*Objects and Controls* のセクション 3.237 「ReturnsNullWhenError」を参照してください。

3.4.6 新しい CompressorObject と ExtractorObject オブジェクト

CompressorObject および ExtractorObject と呼ばれる 2 つの新しいオブジェクトが追加されました。フォルダー、ファイル、あるいはバイトデータストリームを圧縮、または展開できます。サポートされている圧縮形式には、ZIP、7ZIP、GZIP、および TAR が含まれ、ZIP および 7ZIP 形式は、パスワードの AEM-256 暗号化をサポートします。 サポートされている抽出形式には、ZIP、7ZIP、RAR、GZIP、TAR、LZMA、および LZMA86 が含まれます。

これらのオブジェクトの詳細については、*Objects and Controls* のセクション 2.9 「CompressorObject object」および *Objects and Controls* のセクション 2.33 「ExtractorObject オブジェクト」を参照してください。

ファイルまたはデータストリームを圧縮する構文については、*PowerScript Reference* のセクション 2.4.85「Compress」を参照してください。 圧縮されたアーカイブまたはデータストリームを抽出する構文については、*PowerScript Reference* のセクション 2.4.165「Extract」を参照してください。 HTTPClient オブジェクト、RESTClient オブジェクト、または OAuthClient オブジェクトを使用したファイルの圧縮と抽出のコード例については、*Application Techniques* のセクション 4.7.3「Compressing and extracting data」を参照してください。

3.4.7 新しい Import および Export JSON 関数

JSON 文字列とデータウィンドウコントロール、データストアオブジェクト、またはデータウィンドウチャイルドオブジェクトの間で単一のデータ行をインポート / エクスポートするために、次の関数が追加されています：

- ImportRowFromJson：JSON 文字列からデータ行をデータウィンドウコントロール、データストアオブジェクト、またはデータウィンドウチャイルドオブジェクトに挿入します。

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

- ExportRowAsJson：データ行をデータウィンドウコントロール、データストアオブジェクト、またはデータウィンドウチャイルドオブジェクトから JSON 文字列にエクスポートします。

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

詳細については、*DataWindow Reference* のセクション 9.99「ImportRowFromJson」および *DataWindow Reference* のセクション 9.30「ExportRowAsJson」を参照してください。

3.4.8 新しい JSON フォーマット

SnapObjects ModelStore は JSON 文字列で PowerBuilder データウィンドウとデータ

を交換できるため、次の JSON 形式がサポートされるようになりました：

- Plain JSON (以前はシンプル JSON と呼ばれていました)
- DataWindow JSON (以前は標準 JSON と呼ばれていました)
- ModelStore JSON (バージョン 2019 で追加) 注**：バージョン 2019 R2 で削除されました**

これらの形式の詳細については、*Application Techniques* のセクション 4.7.1「Supported JSON formats」を参照してください。

3.5 新しい Windows 10 スタイルのアイコンおよび縮小画像

コントロールの [プロパティ] タブの [アイコン] リストと [縮小画像] リストで選択できる Windows 10 スタイルのアイコンのセットが用意されています。開発者は、アイコン名に追加されたテキストに従って、これらの新しい Windows 10 アイコンをすばやく見つけることができます。たとえば、アイコン名に「_icon_2」が追加され、縮小画像名に「_2」が追加されます。

3.6 64-bit 機能強化

PowerBuilder 2019 では 64-bit のサポートが強化されたため、次のバグはなくなりました：

1. 以前は、開発者がレジストリ関数 (RegistryDelete、RegistryKeys、RegistryGet、RegistrySet、および RegistryValues を含む) を使用して 64-bit オペレーティングシス

テムの 64-bit レジストリエントリにアクセスすると、Wow6432Node レジストリエントリに誤ってリダイレクトされていました。今は正しいレジストリエントリにアクセスできます。

また、これをサポートするために、RegistryGet 関数と RegistrySet 関数に longlong 型列挙値 (RegLongLong!) が追加されています。詳細については、*PowerScript Referenc* のセクション 2.4.617「RegistryGet」または *PowerScript Reference* のセクション 2.4.619「RegistrySet」を参照してください。

2. 以前は、32-bit の PowerBuilder アプリケーションと 64-bit の PowerBuilder アプリケーションは、ランタイムファイルを格納するために同じ場所を使用していたため、同じクライアントマシン上で共存できませんでした。 現行のバージョンでは異なる場所を使用しているので、同じマシン上で共存することができます。

3.7 PBC 機能強化

PBC (PowerBuilder コンパイラ) は、PBL の PBD / DLL ファイルを生成するかどうかを指定できる新しいパラメーター「/pd」をサポートしています。

3.8 新しいオンラインインストーラー - Appeon Installer

PowerBuilder 2019 から、Appeon 製品をインストールするための新しい、より効率的な方法が導入されました。Appeon Installer は、インストールプロセスを案内する自己展開型ダウンロードを備えたオンラインインストーラーです。 インストールプロセス中に、マシンがインターネットに接続する必要があります。

Appeon Installer の使用方法の詳細については、*Installation Guide* のセクション 3.1「Online Installation」を参照してください。

3.9 分類された機能

PowerBuilder に新しい機能が追加されていますが、既存のいくつかの機能は不要または使用が推奨されなくなりました。PowerBuilder Help 内では、次の 3 つの機能分類で表示されています。

- **Discontinued (廃止)** – この機能は、製品から完全に削除されています。

  たとえば、EAServer プロジェクト / ターゲット、PowerBuilder .NET IDE、および Windows Form プロジェクト / ターゲットは廃止された機能です。

- **Obsolete (廃止予定)** – 利用可能ですが、テクニカルサポート対象外で今後の拡張はありません。

  たとえば、Web データウィンドウ、SOAP クライアント、OData サービスと SOAP Web サービスを使用するデータウィンドウデータソース、.NET Web サービスターゲット、.NET アセンブリターゲットは廃止予定の機能です。

- **Stable (安定)** – 利用可能でテクニカルサポートの対象ですが、機能強化は行われません。

4 PowerBuilder 2019 R3 のバグ修正と既知の問題

https://docs.appeon.com/pb2019r3/release_bulletin_for_pb/ を参照してください。

5 PowerBuilder 2019 R2 のバグ修正と既知の問題

PowerBuilder 2019 R2 のバグ修正と既知の問題は、PowerBuilder Release Bulletin の次のリンクにリストされています。

PowerBuilder 2019 R2 のバグ修正: https://docs.appeon.com/pb2019r2/release_bulletin_for_pb/bug_fixes.html

PowerBuilder 2019 R2 の既知の問題:

https://docs.appeon.com/pb2019r2/release_bulletin_for_pb/known_issues.html

6 PowerBuilder 2019 のバグ修正と既知の問題

PowerBuilder 2019 のバグ修正と既知の問題は、PowerBuilderRelease Bulletin の次のリンクにリストされています。

PowerBuilder 2019 のバグ修正:
https://docs.appeon.com/pb2019/release_bulletin_for_pb/ bug_fixes.html

PowerBuilder 2019 の既知の問題:

https://docs.appeon.com/pb2019/release_bulletin_for_pb/known_issues.html