What's New

Appeon PowerBuilder® 2022 R3 (日本語版)

Contents

1 PowerBuilder 2022 R3 新機能 (日本語版)

1.1 新機能

この章について

この章では PowerBuilder 2019 R3 (日本語版) と比較した、PowerBuilder 2022 R3 (日本語版) の新機能を紹介します。

1.1.1 システム要件の更新

- Windows 11 への IDE (PowerBuilder、InfoMaker、SnapDevelop) のインストールをサポートします。

  Windows 8.1 への IDE のインストールは**サポートされなくなりました。**

- Windows 11 および Windows Server 2022 で実行中のアプリ (PowerBuilder アプリケーション プロジェクト、PowerClient プロジェクト、PowerServer プロジェクト、SnapDevelop からデプロイされたものを含む) と PowerServer Web API をサポートします。

  Windows 8.1 および Windows Server 2012 R2 でのアプリの実行は**サポートされなくなりました。**

- 新しくサポートされるデータベース：
- Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) 19.0 & 19.3
- SQL Server 2022 (MSOLEDBSQL 19.3, TLS 1.3 をサポート)
- Informix 14.x
- Oracle 21c
- Linux 上の Oracle 23ai (Oracle Instant Client 21c を使用して接続できます)
- PostgreSQL 15 & 16 (ODBC)

1.1.2 PowerServer の新機能

PowerBuilder の新機能および廃止 / 廃止された機能は、PowerServer にも適用されます。

PowerServer の固有機能については、PowerServer > What's New のオンラインヘルプを参照してください。

1.1.3 PowerClient の新機能

以下は、PowerClient 固有の新機能です。

1.1.3.1 URL 経由で画像を読み込む

画像は、ファイル パスだけでなく、URL アドレス経由でも読み込むことができます。現在は、http および https で始まる URL のみがサポートされています。

ただし、すべてのプロジェクトタイプが URL 経由のイメージの読み込みをサポートしているわけではなく、**PowerClient および PowerServer でデプロイされたアプリケーションのみがこれをサポートします。**

また、すべてのオブジェクト / コントロールが URL 経由で画像を読み込むことができるわけではありません。以下にリストされているオブジェクト / コントロールのみが読み込み可能です(以下にリストされていないオブジェクト / コントロールでも、ファイルパス経由で画像を読み込むことはできます) :

表 1.1:

オブジェクト/ コントロール	プロパティ/関数	使用方法
DataWindow オブジェクト > ボタン コントロール	Filename プロパティ	ペインターの場合 : コントロールの [プロパティ] ビュー > [全般] ページ > [画像ファイル] フィールドに、画像ファイルの URL を入力します。 スクリプトの場合 : dw_1.Object.b_1.filename = "http://172.16.5.175/button2.jpg"

DataWindow オブジェクト > カラム コントロール	BitmapName プロパティ	ペインターの場合：コントロールの [プロパティ] ビュー > [全般] タブで、[画像として表示] オプションを選択し、下の行に画像ファイルの URL を入力します。 スクリプトの場合： dw_1.setitem(1,2,"http://172.16.5.175/button2.jpg")
DataWindow オブジェクト> 計算 フィールドコントロール	Expression プロパティ	ペインタの場合：コントロールのプロパティ ビュー > 一般ページ > 計算式フィールドに、ビットマップ式関数と画像ファイルの URL を入力します (例：Bitmap("http://172.16.5.175/computed2.jpg") スクリプトの場合： dw_1.Object.compute_1.Expression = "bitmap('http://172.16.5.175/computed2.jpg')"
DataWindow オブジェクト> 画像コントロール	Filename プロパティ	ペインターの場合：コントロールのプロパティ ビュー > 全般ページ > ファイル名 フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：dw_1.Object.p_1.filename = "http://172.16.5.175/picture2.jpg"
画像コントロール	PictureName プロパティ	ペインターの場合：コントロールのプロパティ ビュー > 全般ページ > ファイル名 フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：p_1.PictureName = "http://172.16.5.175/tang.jpg"
PictureButton コントロール	PictureName プロパティ	ペインターの場合：コントロールのプロパティ ビュー > 全般ページ > PictureName フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：pb_1.PictureName = "http://172.16.5.175/tang.jpg"

ピクチャーハイパー コントロール	linkPictureName プロパティ	ペインターの場合：コントロールのプロパティ ビュー > 全般ページ > PictureName フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：phl_1.PictureName = "http://172.16.5.175/tang.jpg"

オブジェクト/コントロール	プロパティ/関数	使用方法
PictureListBox コントロール	AddPicture (picturename 関数)	スクリプトの場合：li_pic = plb_1.AddPicture("http://172.16.5.175/picturelistbox3.jpg")
	PictureName[ ] プロパティ	ペインターの場合：コントロールのプロパティ ビュー > 画像タブ > 画像名フィールドに、画像ファイルの URL を入力します。
DropdownPicture コントロール	ListBoxAddPicture (picturename 関数)	スクリプトの場合：li_pic = ddplb_1.AddPicture("http://172.16.5.175/dropdownpicturelistbox2.jpg")
	PictureName[ ] property	ペインターの場合：コントロールのプロパティ ビュー > 画像タブ > 画像名フィールドに、画像ファイルの URL を入力します。
タブコントロール > TabPage	PictureName プロパティ	ペインターの場合：ユーザー オブジェクトの [プロパティ] ビュー > [TabPage ページ] > [PictureName] フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：tab_1.tabpage_1.PictureName = "http://172.16.5.175/tabpage2.jpg"
メニュー オブジェクト > サブメニュー アイテム	MenuImage プロパティ	ペインターの場合：サブメニュー項目のプロパティ ビュー > 一般ページ > MenuImage フィールドに、画像ファイルの URL を入力します。 スクリプトの場合：

		m_webpicture.m_test.m_test1.MenuImage = "http://172.16.5.175/backgroud.bmp"
	ToolbarItem と ToolbarItem プロパティ	ペインターの場合：サブメニュー項目のプロパティビュー > ツールバーページ > ツールバー項目名フィールドとツールバー項目 DownName DownName フィールドに画像ファイルの URL を入力します。 スクリプトの場合： m_webpicture.m_test.m_test1.ToolbarItemName = "http://172.16.5.175/backgroud.bmp"
ListView コントロール	LargePicture プロパティ	ペインターの場合: ListView コントロールのプロパティ ビュー > 大きい画像ページ > LargePictureName フィールドに、画像ファイルの URL を入力します。
	AddLargePicture (picturename 関数)	スクリプトの場合： lv_1.AddLargePicture("http://172.16.5.175/backgroud.bmp")
	SmallPicture プロパティ	ペインターの場合: ListView コントロールのプロパティ ビュー > 小さい画像ページ > 小さい画像の名前フィールドに、画像ファイルの URL を入力します。
	AddSmallPicture (picturename 関数)	スクリプトの場合： lv_1.AddSmallPicture("http://172.16.5.175/listview.jpg")
	StatePicture プロパティ	ペインターの場合：ListView コントロールのプロパティ ビュー > 状態ページ > StatePictureName フィールドに、画像ファイルの URL を入力します。
オブジェクト/コントロール	**プロパティ/関数**	**使用方法**

	AddStatePicture (picturename 関数)	スクリプトの場合： integer index listviewitem lvi_1 lv_1.GetItem(lv_1.SelectedIndex (), lvi_1) index = lv_1.AddStatePicture("http://172.16.5.175 / listview.jpg") lvi_1.StatePictureIndex = index lv_1.SetItem(lv_1.SelectedIndex (), lvi_1)
TreeView コントロール	PictureName[ ] プロパティ	ペインターの場合：TreeView コントロールのプロパティ ビュー > 画像ページ > PictureName フィールドに、画像ファイルの URL を入力します。
	AddPicture (picturename) 関数	スクリプトの場合： long ll_tvi integer li_pic li_pic = tv_1.AddPicture("http://172.16.5.175/ treeview.jpg") ll_tvi = tv_1.FindItem(RootTreeItem!, 0) tv_1.InsertItemFirst(ll_tvi, "New", li_pic)
	StatePicture プロパティ	名前ペインターの場合：TreeView コントロールのプロパティ ビュー > 状態ページ > StatePictureName フィールドに、画像ファイルの URL を入力します。
	AddStatePicture (picturename 関数)	スクリプトの場合： integer index treeviewitem tv_item1 index = tv_1.AddStatePicture("http://172.16.5.17

		5/ treeview.jpg") tv_item1.StatePictureIndex = index

URL 経由で読み込まれた画像に対して、次の設定を取得および設定できます：

- CloudAppSet – URL 経由で読み込まれた画像に対して次の設定を構成します PowerServer / PowerClient でデプロイされたアプリケーション：1) 画像キャッシュ ディレクトリ、2) アプリケーションを閉じるときにキャッシュから画像を削除するかどうか、3) 画像の更新を確認するかどうか。詳細については、CloudAppSet を参照してください。

```
CloudAppSet ("picturecachepath", "/picturecache");
CloudAppSet ("clearpicturecacheonclose", "true");
CloudAppSet ("checkpictureforupdate", "true");
```

- CloudAppGet – PowerServer / PowerClient でデプロイされたアプリケーションで URL 経由でロードされた画像について、1) 画像キャッシュ ディレクトリ、2) アプリケーションを閉じるときにキャッシュから画像を削除するかどうか、3) 画像の更新を確認するかどうかの設定を取得します。詳細については、CloudAppGet を参照してください。

```
CloudAppGet ("picturecachepath", strValue1);
CloudAppGet ("clearpicturecacheonclose", strValue2);
CloudAppGet ("checkpictureforupdate", strValue3);
```

注意

1) 画像ダウンロードの再試行間隔は 30 秒、再試行回数は 3 回です。

2) 画像を正常にダウンロードできない場合は、現在のアプリケーション ディレクトリの “**Log**” フォルダーに **webpicture.log** ファイルが生成されます。

1.1.3.2 署名の改善

1.1.3.2.1 トークンベースの証明書をサポート

アプリケーションをデプロイするときに、トークンベースの証明書 (PFX 証明書に加えて) を使用して、次の実行可能ファイルにデジタル署名できます：

- PowerClient / PowerServer にデプロイされたアプリケーション実行ファイル
- Cloud app launcher

トークンベースの証明書による署名を構成するには、プロジェクト ペインターで**証明書の種類**を **Token-based** に設定し、SignTool の場所、証明書の拇印、署名アルゴリズム、およびタイムスタンプ サーバーの URL を指定する必要があります。詳細については、ユーザー ガイドのセクション 7.3.7.4.5「署名ページ」を参照してください。

1.1.3.2.2 動的パラメーターサポート

スクリプト ファイルで署名する場合、パラメーターを動的に渡すことができるようになりました。

たとえば、次のようなスクリプトを含む cmd ファイルを作成できます：

```
%1 sign /a /fd sha256 /sha1 %2 /tr %3 /td sha256 %4
```

次に、[**独自の署名スクリプトを使用**] フィールドに、cmd ファイルのファイル パスとパラメーター値を入力します (たとえば、Microsoft の SignCode を使用します)：

```
"D:¥2022R3¥test¥testparam.cmd" "D:¥2022R3¥test¥signcode.exe"
"13731a37233bbd83eeb13e95c7898d1d76a2256c" "http://timestamp.digicert.com"
 "salesdemo_local.exe"
```

詳細については、ユーザーガイドのセクション 7.3.7.4.5「署名ページ」を参照してください。

1.1.3.3 コマンドライン引数保存オプション (pbapp.ini 内)

コマンドライン引数がクライアントの URL またはデスクトップ ショートカットで指定され、将来のアクセスのために引数を pbapp.ini に保存する場合は、PowerClient プロジェクト ペインター > スタートアップ ページ > 詳細タブで [**今後のアクセスに備えて引数を pbapp.ini に保存する**] オプションを選択できます。

デフォルトでは、引数は pbapp.ini ファイルに保存されません。

1.1.3.4 CloudAppGet 経由でデプロイメント バージョンを取得する

CloudAppGet 関数は、PowerClient / PowerServer にデプロイされたアプリケーションのデプロイ バージョン (1.01、1.02 など) を取得できます。

```
CloudAppGet ("deploymentversion", strValue4);
```

1.1.4 新しい PDF Builder

新しい PDF Builder は、テキスト、グラフィック、画像、透かしなどのインタラクティブな要素を含む PDF ドキュメントを生成および操作するためのオブジェクトと関数の完全なセットを提供します。

1. PDF Builder で利用できる機能は次のとおりです：

   - PDF ドキュメントを新しい PDF ドキュメントにインポートします；
   - DataWindow、DWChild、または DataStore からデータをインポートします；
   - PDF ドキュメントの結合；
   - PDF ドキュメントのプロパティとセキュリティの設定；
   - 複数行テキスト、単一行テキスト、リッチテキスト、または画像を追加して、入力可能な PDF ドキュメントを作成します；
   - PDF ドキュメントの圧縮；
   - 既存の PDF に単一ページを挿入する；
   - PDF ドキュメント内にドキュメントの他の領域を指すリンクを作成する (たとえば、ドキュメント内に関連するページにリンクするページ番号を追加する)；
   - テキストまたは画像の透かしを追加する；
   - PDF にファイルを添付する

2. PDF Builder は次のオブジェクトを提供します：

   - PDFObject

     PDFObject は、他のすべての PDF Builder オブジェクトの祖先オブジェクトです。PDF Builder オブジェクトの完全な階層を表示するには、ブラウザーで「pdfobject」を見つけてオブジェクトを右クリックし、ポップアップ メニューで「階層を表示」を選択します。

   - PDFObject には、PDFContext、PDFDocExtractor、PDFModel オブジェクトという直接の子オブジェクトがあります。

- PDFModel オブジェクトの名前が PDFModelObject から変更されたことに注意してください。
- PDFModel には次の直接の子オブジェクトがあります：

  PDFAttachment、PDFColor、PDFContent、PDFDocument、PDFDocumentProperties、PDFFont、PDFPage、PDFSecurity、PDFTableOfContents、PDFTableOfContentsItem、PDFTextBlock、PDFTextLayout、PDFWatermark;
- PDFContent には、PDFInvisibleContent および PDFVisibleContent という直接の子オブジェクトがあります；
- PDFVisibleContent には、PDFImage、PDFImportContent、PDFMultilineText、PDFRichText、PDFSharedText、PDFText という直接の子オブジェクトがあります。
- 各オブジェクトのリンクをクリックすると、そのプロパティと機能が表示されます。

注意

PDF ドキュメントの圧縮レベル設定は、主にドキュメント内の高解像度画像に影響します。テキストの圧縮の違いはほとんどありません。

注意

PDF Builder オブジェクト (PDFPage オブジェクトの追加など) を PDF ドキュメントに追加する場合、オブジェクトは 1 回しか追加できません。再度追加する場合は、複製してから追加してください。

---

3. 以下の列挙データ型は PDF Builder に固有のものです：

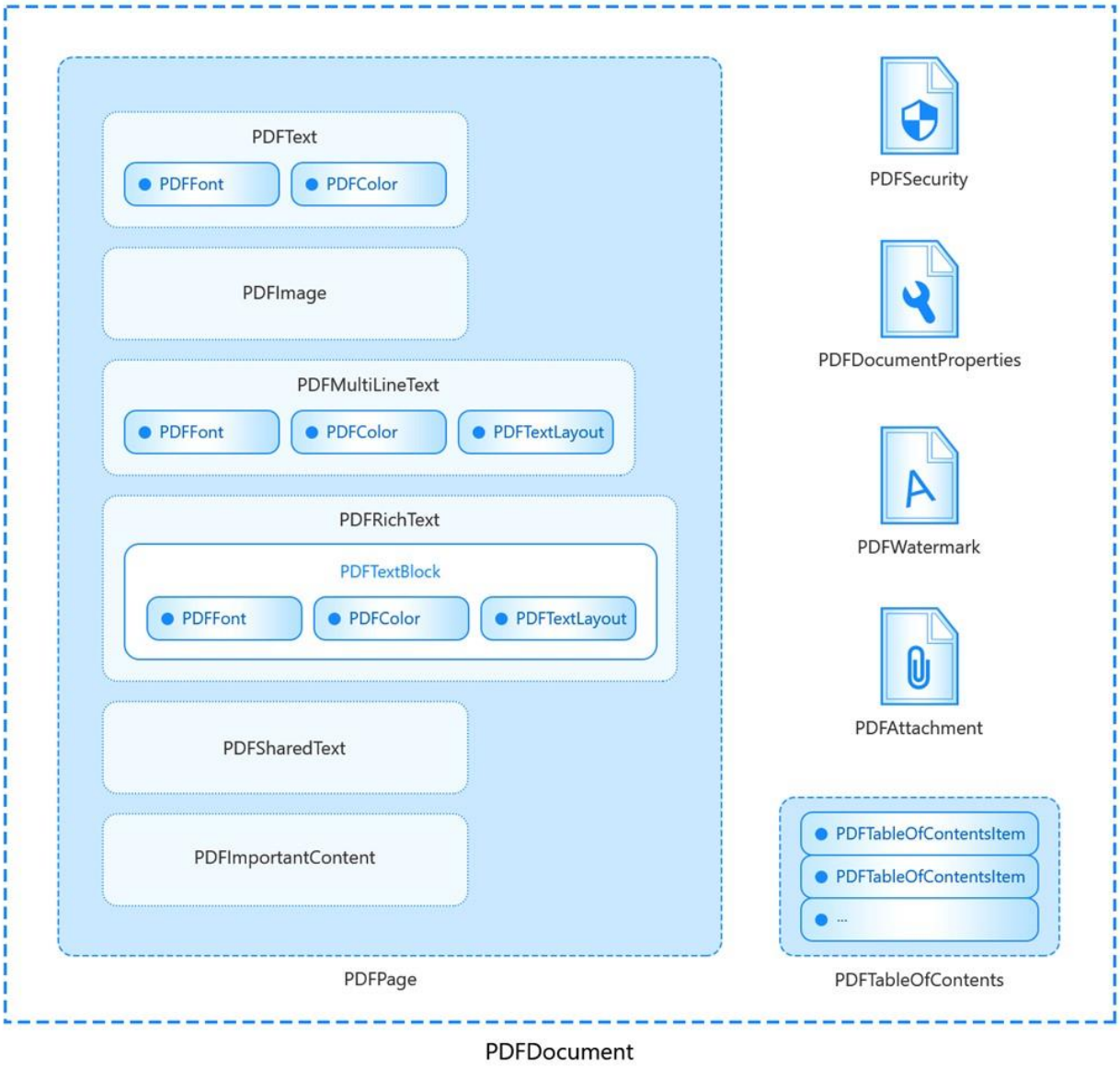
- PDFStandard：PDF_None!、PDFA_1a!、PDFA_1b!、PDFA_3a!、PDFA_3b!、PDFA_3u!

- PDFPageSize：PDFPage_A1!、PDFPage_A2!、PDFPage_A3!、PDFPage_A4!、PDFPage_Letter!、PDFPage_Legal!
- PDFTextAlignment：PDFTextAlignLeft!、PDFTextAlignCenter!、PDFTextAlignRight!、PDFTextAlignJustify!
- PDFImageFitMethod：PDFImageFitmethod_Clip!、PDFImageFitmethod_Entire!、PDFImageFitmethod_Meet!
- PDFTableOfContentsStyle： PDFTableOfContentsStyle_Classic!、PDFTableOfContentsStyle_Simple!
- PDFDocInformation：PDFDocInfo_All!、PDFDocInfo_Properties!、PDFDocInfo_Security!

4. PDF Builder のサポートのために、次のランタイム ファイルが追加されました：**pdflib.dll、pbpdfbuilder.dll、NotoSans-Bold.ttf、NotoSans-Regular.ttf**

各 PDFDocument オブジェクトは PDF ドキュメントを表します。次の図は PDF ドキュメントの構成を示しています：

図 1.1：



製品に含まれるサンプル セールス アプリには、PDF Builder 機能を使用して PDF ドキュメントを作成するサンプル ユーザー ケースが含まれています。興味がある場合は、アプリケーションを実行し、統計ウィンドウを開いて、リボン バーの PDF の作成ボタンを選択してください。

最新のドキュメントについては、「アプリケーションテクニック」のセクション 8.4「アプリケーションでの PDF ドキュメントの作成」を参照してください。

1.1.5 MDI ウィンドウの TabbedView サポート

タブ ビューでは、MDI フレーム ウィンドウで開かれたシートがタブとして表示されます。ウィンドウ タイプが MDI または MDIHelp の場合、次の IDE オプションを直接選択してタブ ビューを実装できます：

- **TabbedView** オプション – MDI クライアント領域の上部に内部タブ付きバー コントロールを追加し、シートをタブ ページとして自動的に表示します。メイン ウィンドウの高さは、タブ付きビューを表示するために自動的に調整されます。
- **MaximizeAllTabbedSheets** オプション – **OpenSheet** / **OpenSheetWithParm** メソッドの *arrangeopen* 引数を無視して、タブ付きシート (**OpenSheet** または **OpenSheetWithParm** 経由で開かれたもの) をすべて最大サイズで表示します。

注意

- **TabbedView** および **aximizeAllTabbedSheets** オプションは、IDE でのみ設定 (有効化または無効化) できます。スクリプト経由で設定することはできません。ただし、IsTabbedViewEnabled および IsMaximizeAllTabbedSheetsEnabled 関数を使用して、MDI ウィンドウで TabbedView および MaximizeAllTabbedSheets オプションが有効になっているかどうかを確認できます。
- **TabbedView** が有効になっている場合、MDI ウィンドウの上に内部タブ付きバー コントロールが追加され、自動的に **mditbb_1** という名前が付けられます。コードがウィンドウ内のすべてのコントロールの存在を確認する場合 (たとえば、コードが PFC フレームワークの "pfc_n_cst_security" オブジェクトの "of_setcontrolstatus" メソッドを呼び出す場合)、エラーを回避するために、**mditbb_1** を確認するようにコードを変更するか、PFC フレームワークをこのバージョン以降にアップグレードしてください。
- タブ付きバー コントロールには関数やイベントはありませんが、スクリプトを使用してコントロールのプロパティ (アイコン、テキスト、PowerTips、コンテキスト メニュー、タブ付きバーの高さ、タブの位置など) を取得および設定できます。ここで例を参照してください。
- **MaximizeAllTabbedSheets** オプションが選択されている場合、シート ウィンドウのサイズ変更イベントが 2 回トリガーされます。最初のトリガーはシート ウィンドウが開かれたときに発

生し、2 番目のトリガーは **MaximizeAllTabbedSheets** オプションが選択されていることが検出されたときに発生します。

- 他のコントロールと同様に、タブ付きバー コントロールをウィンドウに手動で追加 (および操作) する (たとえば、メニューの [挿入] > [コントロール] から) ことは、将来のバージョンでサポートされる予定です。
- タブ ビューが機能するには、次のランタイム ファイルが必要です：**pbtabbedbar.dll**

詳細については、「アプリケーションテクニック」のセクション 3.1.4「タブビューでシートを表示する」を参照してください。

1.1.6 PBAutoBuild ツール

PBAutoBuild220.exe は、Windows で DOS コマンドを使用してプロジェクトをコンパイルおよび展開できるスタンドアロン ツールです。PowerBuilder IDE をインストールしたり、PowerBuilder ライセンスを実行しなくても、ビルドと展開のプロセス全体を無料で使用でき、自動化できます。PowerBuilder IDE でプロジェクトをビルドするだけでなく、コマンド ライン ツール PBAutoBuild220.exe を使用してプロジェクトをビルドすることもできます。Jenkins やその他のフレームワークを使用して CI/CD パイプラインを作成することもできます。

PBAutoBuild220.exe を使用して PowerServer プロジェクトをビルドする方法については、「コマンドを使用した PowerServer プロジェクトのビルド」および「チュートリアル：コマンドラインを使用した PowerServer プロジェクトのビルド」を参照してください。

PBAutoBuild220.exe を使用して PowerClient プロジェクトをビルドする方法については、ユーザー ガイドの「コマンドを使用した PowerClient プロジェクトのビルド」および「チュートリアル：コマンドを使用した PowerClient プロジェクトのビルド」を参照してください。

PBAutoBuild220.exe を使用してアプリケーション プロジェクトをビルドする方法については、ユーザー ガイドの「コマンドを使用したクライアント/サーバー アプリケーションの構築」および「チュートリアル：コマンドを使用したクライアント/サーバー アプリケーションの構築」を参照してください。

1.1.7 ネイティブ電子メールサポート (SMTP クライアント)

ネイティブ電子メールのサポート用に、次の機能を備えた 2 つの新しいオブジェクト SMTPClient と MimeMessage が追加されました：

- 電子メール サービス用の電子メール サーバーとの直接のやり取りを可能にします。

- SMTP および SMTPS プロトコルをサポートします。
- HTML およびプレーンテキスト形式をサポートします。
- 次のいずれかの接続タイプの使用をサポートします：なし (プレーン テキスト)、STARTTLS、および Auto STARTTLS。
- 次の認証タイプをサポートします ：なし、自動、CRAM-MD5、LOGIN、PLAIN、NTLM、および XOAUTH2。XOAUTH2 認証の場合、SMTPClient オブジェクトの XOAUTH2 アクセス トークンを指定できます。その他の認証タイプは、クライアントがサーバーに接続するときに自動的に検出されます。
- 非同期モードでの電子メールの送信をサポートします。
- プロキシ サーバー経由の電子メール送信をサポートします。

ネイティブ電子メールのサポートには、次のランタイム ファイルが必要です：**pbsmtpclient.dll**、**libcurl.dll**。

最新のドキュメントについては、アプリケーション テクニックのセクション 5.4.2「SMTP について」を参照してください。さまざまなシナリオで SMTP を使用するためのサンプル スクリプトについては、アプリケーション テクニックのセクション 5.4.3「SMTP の使用」を参照してください。

1.1.8 リッチテキストエディタと RichTextEdit コントロールの機能強化

1.1.8.1 テキストコントロールの変更

- **組み込みのリッチ エディット コントロール (TE エディット コントロール)** は削除されました。アプリケーション テクニックの TX テキスト コントロールに置き換えることができます。
- **組み込みの TX Text Control ActiveX 15.0** および **TX Text Control ActiveX 24.0 Professional/Enterprise** は削除されました。
- リッチ テキスト エディターは以前、 PDF 結合機能のためにサードパーティ ソフトウェア Podofo と連携していましたが、現在は PDFlib と連携するように変更されています。
- TX テキスト コントロール ActiveX は、32-bit と 64-bit の両方で最新バージョン (バージョン 31) にアップグレードされました。また、ランタイム ファイルもバージョン 31 で動作するように更新

されました。「アプリケーション テクニック」のセクション 9.2.4「PowerBuilder ランタイム ファイル」を参照してください。

リッチ テキスト エディターのツールバーには、次の新しいボタンがあります：

- RightToLeft – 右から左の順序で文字を挿入して表示します。
- LeftToRight – 左から右の順序で文字を挿入して表示します。
- Increase indent – テキストまたは段落をさらに右にインデントします。
- Decrease indent – テキストまたは段落を左余白に近づけます。

1.1.8.2 RichTextEdit コントロールの新機能

RichTextEdit コントロールには次の機能強化が加えられています：

- フォーム フィールドをサポートします。サポートされるフォーム フィールドの種類は次のとおりです：FormFieldCheckBox、FormFieldComboBox、FormFieldText、および FormFieldDate

---

注意

SaveDocument 関数を実行して RichTextEdit コントロールのコンテンツを PDF ファイルに保存すると、フォーム フィールドは PDF ファイル内で編集可能なままになります。

- フォーム フィールドの種類に関係なく、フォーム フィールドに次の新しい関数が追加されます：

  FormFieldGetEmptyWidth; FormFieldSetEmptyWidth;
  FormFieldGetText;
  FormFieldSetText; FormFieldGetCurrent; FormFieldSetCurrent;
  FormFieldGetDeletable; FormFieldSetDeletable; FormFieldDelete;
  FormFieldGetEnd; FormFieldGetStart; FormFieldNext

- フォーム フィールド タイプ FormFieldCheckBox に次の新しい関数が追加されました：

  FormCheckBoxInsert; FormCheckBoxGetChecked;

  FormCheckBoxSetChecked

- フォーム フィールド タイプ FormFieldComboBox に次の新しい関数が追加されました：

  FormComboBoxInsert; FormComboBoxGetItems;

  FormComboBoxSetItems

- フォーム フィールド タイプ FormFieldText に次の新しい関数が追加されました：

  FormTextFieldInsert

- フォーム フィールド タイプ FormFieldDate に次の新しい関数が追加されました：

  FormDateFieldInsert; FormDateFieldGetDate; FormDateFieldSetDate;
  FormDateFieldGetFormat; FormDateFieldSetFormat

- 表をサポートします。

- 表の追加 / 削除、表の列または行の追加のために、次の新しい関数が追加されました：

  TableInsert; TableInsertColumn; TableInsertDialog; TableInsertRows;
  TableDelete; TableDeleteColumn; TableDeleteColumns;
  TableDeleteRow; TableDeleteRows

- 表内のセルを操作するための次の新しい関数が追加されました：

  TableCanChangeAttr; TableCellSelect; TableCellStart;
  TableGetCellBackColor;
  TableSetCellBackColor; TableGetCellBorderColor;
  TableSetCellBorderColor;
  TableGetCellBorderWidth; TableSetCellBorderWidth;
  TableGetCellHeader;
  TableSetCellHeader; TableGetCellHeight; TableSetCellHeight;
  TableGetCellHorizontalExt; TableSetCellHorizontalExt;
  TableGetCellHorizontalPos;

TableSetCellHorizontalPos; TableGetCellLength;
TableGetCellNumberFormat;
TableSetCellNumberFormat; TableGetCellText; TableSetCellText;
TableGetCellTextGap; TableSetCellTextGap; TableGetCellTextType;
TableSetCellTextType; TableGetCellVertAlign; TableSetCellVertAlign;
TableMergeCells; TableSplitCells

- 指定された表、行、または列を検索するための次の新しい関数が追加されました：

  TableAtInputPos; TableColumnAtInputPos; TableFromSelection;
  TableNext; TableRowAtInputPos

- 表プロパティを管理するための次の新しい関数が追加されました：

  TablePropertiesDialog; TableGetColumnCount; TableGetRowCount

- テキストフレームをサポートします。

- テキスト フレームを挿入または選択するための次の新しい機能が追加されました：

  TextFrameInsert; TextFrameInsertAsChar; TextFrameInsertFixed;
  TextFrameSelect

- テキスト フレーム内のテキストを設定または取得するための次の新しい関数が追加されました：

  TextFrameGetText; TextFrameSetText

- テキスト フレームのプロパティを管理するための次の新しい関数が追加されました：

  TextFrameGetBackColor; TextFrameSetBackColor;
  TextFrameGetBorderWidth;
  TextFrameSetBorderWidth; TextFrameGetInternalMargin;
  TextFrameSetInternalMargin; TextFrameGetMarkerLines;
  TextFrameSetMarkerLines

- RightToLeft プロパティをサポートします。このプロパティは、IDE ペインターまたはアプリケーション スクリプトで設定できます。

- RichTextEdit コントロールと RichText データウィンドウ オブジェクトでは、RightToLeft = true にすると、テキスト エディターに右から左の順序で文字が挿入され、ツール バー、ステータス バー、ルーラー、ポップアップ メニュー、ダイアログ ボックスでレイアウトが右揃えで表示されます。

- リッチテキスト編集スタイルの列では、RightToLeft = true にすると、アクティブ セルに右から左の順序で文字が挿入され、現在の列のツールバーに右揃えのレイアウトが表示されます。

- PDF/A 標準を使用して、RichTextEdit コントロールの内容を PDF として保存することをサポートします。

- userPassword、masterPassword、制限の設定と、その内容を PDF として保存する機能をサポート

- Microsoft Excel 形式 (.xlsx) のファイルを RichTextEdit コントロールに挿入できます

- "UTF-8"、"UTF-16LE"、"UTF-16BE" などの異なるエンコード形式のテキストファイルを不正なコードなしで正しく挿入できます (BOM ヘッダーまたはエンコードを指定する必要があります)

- クリップボードの内容を指定された形式で貼り付けることをサポートします (PasteSpecial 機能を使用するか、マウスの右ボタン > **Paste Special** ボタンをクリック)
- テキスト内のハイパーリンクのオープン、ハイパーリンクの作成、変更、削除をサポートします

- 以前に元に戻した編集の復元をサポート

RichTextEdit コントロールに次の新しい関数が追加されました：

- Redo – RichTextEdit コントロールで以前に元に戻された編集を復元します。

- CanRedo – Redo が RichTextEdit コントロールで以前に元に戻された編集を復元できるかどうかをテストします。

- PasteSpecial – 指定された形式でクリップボードの内容を RichTextEdit コントロールの現在の位置に貼り付けます。

- TextFieldInsert – RichTextEdit コントロールの現在の位置にテキスト フィールドを挿入します。
- TextFieldSetText – RichTextEdit コントロールのテキスト フィールドのテキストを設定します。
- TextFieldSetTypeAndData – RichTextEdit コントロール内のテキスト フィールドのタイプとタイプ データを設定します。
- TextFieldGetType – RichTextEdit コントロール内の指定されたテキスト フィールドの種類を取得します。
- TextFieldGetText – RichTextEdit コントロール内の指定されたテキスト フィールドのテキストを取得します。
- TextFieldGetTypeData – RichTextEdit コントロール内の指定されたテキスト フィールドのテキストを取得します。
- TargetInsert – RichTextEdit コントロールの現在の位置にターゲットを挿入します。
- TargetDelete – RichTextEdit コントロール内の指定された ID に従ってターゲットを削除します。
- TargetSetName – RichTextEdit コントロール内の指定されたターゲットの名前を設定します。
- TargetGetName – RichTextEdit コントロール内の指定されたターゲットの名前を取得します。
- TargetNext – 指定されたターゲットの次のターゲット ID を取得します。
- TargetGoto – RichTextEdit コントロール内の指定されたターゲットにカーソルを移動します。
- SaveDocumentAsPDF – RichTextEdit コントロールの内容を PDF ドキュメントとして保存します。

以下の機能が変更されました：

- InsertDocument – InsertDocument 関数に、新しい引数 encoding と新しいファイル タイプ FileTypeXLSX! が追加されました。挿入されたドキュメントのエンコードを指定できます。エンコード引数が指定されていない場合は、BOM ヘッダーのエンコードが使用されます。ファイルに BOM ヘッダーがない場合は、ANSI が既定のエンコード形式として使用されます。

  .xlsx ファイル タイプも [**ファイル名の選択**] ダイアログ ボックスに追加されるので、マウスの右ボタン > [**ファイルの挿入**] ボタンをクリックすることで、Microsoft Excel (.xlsx) 形式のファイルを挿入することを選択できます。

  [**ファイル名の選択**] ダイアログ ボックスから .txt ファイルを挿入する場合、BOM ヘッダーのエンコード (UTF8、UTF16LE、または UTF16BE) が使用されます。ファイルに BOM ヘッダーがない場合は、デフォルトで ANSI が使用されます。

1.1.9 テーマの強化

1.1.9.1 テーマのカスタマイズ

標準ビジュアル ユーザ オブジェクトとその子孫ユーザ オブジェクトのテーマ設定のカスタマイズをサポートします。

標準ビジュアル ユーザ オブジェクトのテーマ設定を構成できます。テーマ設定は、その子孫ユーザ オブジェクトすべてにも適用されます。または、個々の子孫ユーザ オブジェクトのテーマ設定を構成します。

- 子孫ウィンドウのコントロール / オブジェクトのテーマ設定のカスタマイズをサポートします (子孫ウィンドウとは、別のウィンドウから継承されたウィンドウのことです) : [子孫ウィンドウ].[コントロールまたはオブジェクト]。
- 子孫ウィンドウのコントロール/オブジェクトにカスタム テーマ設定がある場合は、そのカスタム テーマ設定が有効になります。子孫ウィンドウのコントロール/オブジェクトにカスタム テーマ設定がない場合は、祖先ウィンドウのコントロール/オブジェクトからテーマ設定を継承します。

  子孫ウィンドウのコントロール / オブジェクトにカスタム テーマ設定がある場合は、そのカスタム テーマ設定が有効になります。子孫ウィンドウのコントロール / オブジェクトにカスタム テーマ設定がない場合は、祖先ウィンドウのコントロール / オブジェクトからテーマ設定を継承します。

- 子孫ウィンドウ内の同じタイプのコントロール/オブジェクトのテーマ設定のカスタマイズをサポートします (子孫ウィンドウとは、別のウィンドウから継承されたウィンドウです) : [子孫ウィンドウ]$[コントロールタイプ]

  子孫ウィンドウのコントロール / オブジェクトにカスタム テーマ設定がある場合は、そのカスタム テーマ設定が有効になります。子孫ウィンドウのコントロール / オブジェクトにカスタム テーマ設定がない場合は、祖先ウィンドウの同じタイプのコントロール / オブジェクトからテーマ設定を継承します。

- ユーザ オブジェクト内の同じタイプのコントロール / オブジェクトのテーマ設定のカスタマイズをサポートします : [ユーザオブジェクト]$[コントロールタイプ]

- 子孫ユーザ オブジェクト内の同じタイプのコントロール / オブジェクトのテーマ設定のカスタマイズをサポートします (子孫ユーザ オブジェクトとは、別のユーザ オブジェクトから継承されたユーザ オブジェクトです) : [子孫ユーザ オブジェクト]$[コントロールタイプ]

  子孫ユーザ オブジェクト内のコントロール / オブジェクトにカスタム テーマ設定がある場合は、そのカスタム テーマ設定が有効になります。子孫ユーザ オブジェクト内のコントロール / オブジェクトにカスタム テーマ設定がない場合は、祖先ユーザ オブジェクト内の同じタイプのコントロール / オブジェクトからテーマ設定を継承します。

詳細については、ユーザー ガイドの「カスタム テーマの構成」セクションと「優先順位」セクションを参照してください。

1.1.9.2 テーマのサイズ変更の最適化

UI テーマを適用したウィンドウにはモダンな外観の 2D 境界線がありますが、デフォルトの境界線 (幅 1 ピクセル) ではウィンドウをつかんでサイズを変更するのが非常に困難です。

境界線が再設計され、境界線のサイズと外観を変更せずにグラブ領域のサイズが拡大されました。

- MDI シートの場合

  border = 1 (デフォルト値)、border <= 0、または border > 8 (ピクセル) の場合、境界線の幅は常に 1 ピクセルになり、グラブ領域は 1 ピクセル幅から 4 ピクセル幅に拡大されます。

border >= 2 かつ <= 8 の場合、境界線は指定された数値を幅として使用し、グラブ領域は境界線と同じ幅になります (たとえば、border = 3 の場合、境界線とグラブ領域の両方の幅は 3 ピクセルになります)。

- その他のウィンドウの場合

  border = 1 (デフォルト値)、border <= 0、または border > 8 (ピクセル) の場合、境界線は常に 1 ピクセル幅になり、グラブ領域は 1 ピクセル幅から 6 ピクセル幅に拡大されます。

  border >= 2 および <= 8 の場合、境界線は指定された数値を幅として使用し、グラブ領域は境界線と同じ幅になります。

1.1.9.3 利用可能なテーマの追加

2 つの新しいシステム テーマが追加されました (現在、システム テーマは 6 つあります) :

- Flat Design Lime – このテーマは、Flat Design Blue に似ていますが、配色が緑色です。コントロールはフラットでシンプルなスタイルで設計されており、デフォルトの状態では緑色です。インターフェイスはすっきりとしたモダンな外観で、すべての要素が簡単に区別でき、視覚的に魅力的です。

- Flat Design Orange – このテーマは、Flat Design Blue に似ていますが、オレンジ色の配色になっています。コントロールはフラット化されており、デフォルトの状態ではオレンジ色になっています。インターフェイスは、さまざまな要素間のコントラストがはっきりしており、温かみのある魅力的な雰囲気になります。

1.1.10 PowerBuilder オブジェクトと PowerScript の機能強化

1.1.10.1 WebBrowser コントロールの強化

1.1.10.1.1 WebBrowser エンジンのアップグレード

WebBrowser コントロールのエンジンが Chromium Embedded Framework (CEF) から Microsoft Edge WebView2 に変更されました。

アプリケーションで WebBrowser コントロールを使用する場合は、次の変更に注意してください。

1. 既存関数の変更

   グローバル関数 WebBrowserSet および WebBrowserGet は、WebView2 で動作するように調整されました。

2. 新規関数

   - NavigateToString 関数 – 指定された HTML ソース コードからページに移動して読み込みます。
   - PostJsonWebMessage と PostStringWebMessage 関数 – JSON または単純な文字列で Web メッセージを非同期的に送信します。
   - OpenDefaultDownloadDialog と CloseDefaultDownloadDialog 関数 – デフォルトのダウンロード ダイアログを開いたり閉じたりします。

3. 既存イベントの変更

   次のイベントには追加の引数があります：AddressChange、DownloadingStateChanged、DownloadingStar、EvaluateJavascriptFinished、NavigationStart、PdfPrintFinished.

4. 新規イベント

   - AcceleratorKeyPressed イベント – キーボード操作 (F5 など) への応答をサポートします。
   - その他の新しいイベントには、ContentLoading、DOMContentLoaded、DownloadingOperationStateChanged、EstimatedEndTimeChanged、HistoryChanged、IsDefaultDownloadDialogStateChanged、LoseFocus、NavigationCompleted、WebMessageReceived が含まれます。

5. ランタイムファイルの変更

   WebView2 ランタイムは、Evergreen ランタイムと固定バージョンの 2 つのバージョンをサポートしています。使用する WebView2 ランタイム バージョンについては、ユーザー ガイドのセクション 4.5.2「WebView2 ランタイム バージョンの選択」を参照してください。

- Evergreen Runtime を使用する場合は、**pbwebbrowser.dll** と **webbrowserapi.tlb** が必要です。

- 固定バージョンを使用する場合は、**pbwebbrowser.dll**、**webbrowserapi.tlb**、および "**PBWebView2**" フォルダー内のすべてのファイルが必要です。

製品に同梱されているサンプルグラフアプリには、WebBrowser 上でグラフを読み込むサンプルが含まれています。興味がある場合は、アプリケーションを実行して、さまざまなグラフをご覧ください。最新のドキュメントについては、ユーザー ガイドのセクション 4.5「WebBrowser の操作」およびオブジェクトとコントロールのセクション 2.173「WebBrowser コントロール」を参照してください。

1.1.10.1.2 WebBrowser 上のリモートデバッグのサポート

WebBrowser コントロール (WebView2 ベース) は、指定されたポートでの HTTP 経由のリモート デバッグをサポートするようになりました。

次のスクリプトを使用してリモート デバッグを有効にすることができます：

```
li_return = WebBrowserSet("remote-debugging-port", "8210") li_return = WebBrowserSet("remote-allow-origins", "*")
```

注意：これらのスクリプトは、WebBrowser コントロールを含むウィンドウが開かれる**前**に実行する必要があります。これらのスクリプトは、アプリケーションの Open イベント、または WebBrowser コントロールを含むウィンドウが開かれる前に実行される任意の場所に配置できます。remote-allow-origins および remote-debugging-port 設定は、アプリケーションがデバッグ モードの場合にのみ使用してください。最適なパフォーマンスを得るには、アプリケーションがデバッグ モードではないときにこれらの設定をコメント アウトしてください。

詳細については、WebBrowserSet を参照してください。

1.1.10.2 HTTPClient/RestClient/TokenRequest/OAuthRequest 強化

1.1.10.2.1 TLS 1.3 のサポート

HTTPClient/RestClient/TokenRequest/OAuthRequest オブジェクトは、TLS 1.3 セキュリティ プロトコルの使用をサポートしています：

- SecureProtocol プロパティ – このプロパティ値を 6 に設定すると、TLS 1.3 プロトコルが使用されます。

• リクエスト送信機能は、TLS 1.3 エラーが発生した場合にエラー コードを返します。

現在、PowerBuilder ネイティブ C/S アプリケーションは、Windows 11 または Windows Server 2022 で実行されている場合にのみ TLS 1.3 をサポートできます。

Windows 10 で TLS 1.3 をサポートするには、レジストリ キーを変更する必要があります (非推奨)。

1.1.10.2.2 HTTP/2 のサポート

HTTPClient/RestClient/TokenRequest/OAuthRequest オブジェクトは、HTTP/2 を使用した接続をサポートします：

• EnableHttp2 プロパティ – このプロパティはデフォルトで True に設定されています。つまり、リクエストを送信するときにデフォルトで HTTP/2 接続が使用されます。HTTP/2 接続を使用しない場合は、このプロパティ値を False に設定できます。HTTP/2 が十分にサポートされていない環境では、システムは HTTP/1.1 にフォールバックしてトランザクションを続行します。

HTTPClient/RestClient/TokenResponse/ResourceResponse オブジェクトは、現在使用されている HTTP プロトコル バージョンを取得できます：

• GetHttpVersion 関数 – 通信で使用される HTTP プロトコルのバージョンを取得します。

1.1.10.2.3 双方向 TLS 認証をサポート

HTTPClient/RestClient/TokenRequest/OAuthRequest オブジェクトは、クライアントとサーバー間の双方向 TLS 認証をサポートするために次の機能を提供します：

• SetClientCert – サーバーにアクセスするために使用するクライアント証明書を設定します。

• ClearClientCert – SetClientCert で設定されたクライアント証明書をクリアします。

RestClient/TokenRequest/OAuthRequest オブジェクトは次のプロパティを提供します：IgnoreServerCertificate プロパティと CheckForServerCertRevocation プロパティ。(HTTPClient オブジェクトはバージョン 2019 R3 で既にこれらのプロパティをサポートしています)

クライアントは、HTTPClient/RestClient/TokenRequest/OAuthRequest オブジェクトの次のプロパティを利用して、クライアント証明書を使用せずにサーバーにアクセスすることもできます (匿名アクセス)：

- AnonymousAccess プロパティ – このプロパティ値を True に設定すると、クライアントは匿名でサーバーにアクセスできるようになります。

1.1.10.2.4 認証資格情報の送信をサポート

HTTPClient オブジェクトは、サーバー リソースにアクセスするときに、認証資格情報 (Windows 資格情報など) をサーバー / プロキシに送信できるようになりました。新しく追加された機能は次のとおりです：

- GetSupportScheme：ネゴシエート、NTLM、ダイジェスト、ベーシックなどのサポートされている認証スキームを取得します。
- SetCredentials：必要な認証資格情報を設定し、サーバーに渡します。
- ResendPostRequest：必要に応じてリクエストを再送信します。

1.1.10.2.5 データ抽出をサポート

HTTPClient および OAuthClient オブジェクトは、データを抽出するために ExtractorObject オブジェクトを呼び出す必要がなくなりました。次の関数が強化され、レスポンスデータを gzip または brotli 形式 (レスポンスヘッダーが "Content-Encoding: gzip" または "Content-Encoding: br") で自動的に展開できるようになりました：

- HTTPClient の SendRequest と ReadData
- OAuthClient の RequestResource

1.1.10.3 JSONParser 強化

JSONParser は、新しい関数 GetItemDecimal を使用して小数値の取得をサポートしているため、JSON 文字列を介して値を転送するときに 28 桁の精度をサポートできます。

詳細については、GetItemDecimal を参照してください。

1.1.10.4 PBDOM 強化

PBDOM_Document オブジェクトに次の 2 つの新しい関数が追加されました：

- SetXMLDeclaration – XML ドキュメントの宣言を設定します。
- GetXMLDeclaration – XML ドキュメントの宣言を取得します。

PBDOM_Document オブジェクトに 3 つの新しい引数 (version, encoding, standalone) を導入した後、SaveDocument 関数は次の点で変更されます：

- ファイルは、SaveDocument 関数によって指定されたエンコーディングで保存されます。新しいドキュメントを作成する場合、エンコーディングのデフォルト値は UTF-16LE です。既存のドキュメントを開く場合、エンコーディングはドキュメントの宣言で指定されます。ドキュメントに宣言がない場合、または宣言でエンコーディングが指定されていない場合、エンコーディングはデフォルトで UTF-16LE になります。
- standalone が空の文字列の場合、文字列をディスク ファイルに保存するときにデフォルトで ”no” に設定されます。version が無効な場合、宣言は追加されません。また、encoding が無効な場合、例外はスローされませんが、空のドキュメントが生成されます。

PBDOM_Attribute および PBDOM_PROCESSINGINSTRUCTION オブジェクトは、次の点で変更されました：

- PBDOM_ATTRIBUTE オブジェクトを作成すると、オブジェクトにはデフォルトで子ノードが含まれません (これは、オブジェクトにデフォルトで子ノードが含まれていた以前のバージョンとは異なります) 。
- PBDOM_PROCESSINGINSTRUCTION オブジェクトは、宣言の追加と変更がサポートされなくなりました。PBDOM_PROCESSINGINSTRUCTION のノード名が "XML" に設定されている場合、例外がスローされます。

PowerBuilder 拡張機能リファレンス メソッドの PBDOM_BUILDER SetDisableEntityResolution は、XML ドキュメントで参照される外部エンティティをロードしないことで、ユーザーが XML 外部エンティティ (XXE) 攻撃を防止できるようにするために追加されました。

```
public subroutine  SetDisableEntityResolution(boolean bDisableEntityResolution)
```

True の場合、XML ドキュメントで参照される外部エンティティを読み込むことはできません。
False (デフォルト) の場合、外部エンティティを読み込むことができます。

If PBDOM is used in your application, please follow the instructions in > to upgrade and modify your application.

アプリケーションで PBDOM が使用されている場合は、PowerBuilder アプリケーションのアップグレード > PBDOM の変更 の手順に従って、アプリケーションをアップグレードおよび変更してください。

1.1.10.5 DDDW & DDLB 強化

1.1.10.5.1 新しい dddw.validation および ddlb.validation プロパティ

ユーザーが DropDownDataWindow または DropDownListBox 編集スタイル列に値を入力すると、その値がドロップダウン リストに存在するかどうかがチェックされます。存在する場合は値が受け入れられ、存在しない場合は、ユーザーに別の値を入力するように求められます。

この検証チェックを有効にするには、ペインターで「ドロップダウンの選択肢に対する値」オプションを選択するか、スクリプトで dddw.validation または ddlb.validation プロパティを設定します。このオプション / プロパティは、AllowEdit が有効な場合に有効です。

1.1.10.5.2 DDDW と DDLB のオートコンプリート

AutoCompleteMode プロパティが データウィンドウ オブジェクト プロパティ dddw.property および ddlb.property に追加され、次の機能がサポートされます：

- DropDownDataWindow および DropDownListBox でのデータ入力の自動補完をサポートします。
- DropDownDataWindow および DropDownListBox に入力されたデータに応じてデータをフィルタリングする機能をサポートします。

1.1.10.6 .NET assemblie 呼び出しの機能強化

C# での PowerScript イベントの呼び出しをサポートするために、PowerBuilder DotNetObject オブジェクトに次の 2 つの新しい関数が追加されました：

- RegisterObject – PowerBuilder オブジェクトを登録します。登録後、PowerBuilder オブジェクト内のイベントは .NET assembly によってトリガーできます。.NET assembly によってトリガーされるイベントは、文字列型のパラメータを 1 つだけ持つか、パラメーターを持たず、文字列型の戻り値を 1 つだけ持つか、戻り値を持たないカスタム イベントです。
- UnregisterObject – PowerBuilder オブジェクトの登録を解除します。

PowerScript ユーザー イベントは、次の .NET メソッドを通じて C# スクリプトによってトリガーできます：

- PowerBuilder.RegisteredObject.TriggerEvent – 登録された PowerBuilder オブジェクトでユーザー イベントをトリガーします。ユーザー イベントには、文字列型パラメーターを 1 つだけ持つか、パラメーターを持たず、文字列型戻り値を 1 つだけ持つか、戻り値を持たずに済みます。

.NET assembly からロードされた関数では、ジェネリック型、デリゲート、インターフェイス、または非配列クラス (抽象クラスを含む) をパラメーターまたは戻り値として渡すことができます。(**.NET DLL Importer** にもこれらの拡張機能があります。詳細については、アプリケーション テクニックのこのセクションを参照してください。) PowerBuilder DotNetAssembly オブジェクトに、次の新しい関数が追加されました：

- LoadWithDotNet – NET (5 以降) assembly を読み込みます。

1.1.10.7 新しい OpenURL 関数

新しい OpenURL システム関数は、Inet オブジェクトの HyperLinkToURL 関数と同じ機能を持ちます。

```
Integer OpenUrl( string url )
```

次の理由により、廃止された Inet オブジェクトの使用を完全に停止することが可能です：

- OpenURL 関数は、Inet.HyperLinkToURL 関数を置き換えて、デフォルトの Web ブラウザー (アプリケーションの外部) を開き、指定された URL に移動できます。
- HTTPClient.SendRequest ("GET", ...) は Inet.GetURL 関数を置き換えることができます。

  例えば、

```
Integer li_rc
String ls_string HttpClient lnv_HttpClient
lnv_HttpClient = Create HttpClient

// GET メソッドを使用してリクエストを送信する
li_rc = lnv_HttpClient.SendRequest("GET", "https://demo.appeon.com/PB/
webapi_client/employee/102") // レスポンスデータを取得する
```

```
if li_rc = 1 and lnv_HttpClient.GetResponseStatusCode() = 200 then
lnv_HttpClient.GetResponseBody(ls_string) end if
```

- HTTPClient.SendRequest ("POST", ...) は Inet.PostURL 関数を置き換えることができます。

  例えば、

```
Integer li_rc
String ls_ReturnJson HttpClient lnv_HttpClient lnv_HttpClient = Create HttpClient
String ls_json = '{"empId":100, "fname":" John", "lname": "Guevara"}'

// POST リクエストを構築します（すべてのヘッダーをサポート）

lnv_HttpClient.SetRequestHeader("Content-Type", "application/json;charset=UTF-8") //
SendRequest によって設定される Content-Length ヘッダー

// POST メソッドを使用してリクエストを送信します (文字列データを本文に追加し、Content-Length ヘッダーに設定します)

li_rc = lnv_HttpClient.SendRequest("POST", "https://demo.appeon.com/PB/
webapi_client/employee", ls_json)

// レスポンスデータを取得する
if li_rc = 1 and lnv_HttpClient.GetResponseStatusCode() = 200 then
lnv_HttpClient.GetResponseBody(ls_ReturnJson) end if
```

1.1.11 IDE 強化

PowerBuilder IDE には次の機能強化が加えられています：

- 「最近使用したオブジェクト」メニュー リストからすべてのオブジェクトまたは特定のオブジェクトを削除できるようになりました。

  メニューの [ファイル] > [最近使用したオブジェクト] > [すべてクリア] を選択して、最近使用したオブジェクトを一度にすべて削除するか、各オブジェクトの後の閉じるアイコンをクリックして個々のオブジェクトを削除できます。

- IDE から、ランタイム パッケージャー (IDE でメニュー [ファイル] > [新規] を選択し、[新規] ダイアログの [ツール] タブからツールを選択することによって) を直接起動できるようになりました。

- 「クイックウォッチ」ウィンドウ、「オブジェクトの挿入」ウィンドウ、「更新プロパティの指定」ウィンドウなど、多数のウィンドウ / ダイアログのサイズを変更できます。

- 「検索引数の指定」ダイアログでマウス ホイールを使用してレコードをスクロールできるようになりました。

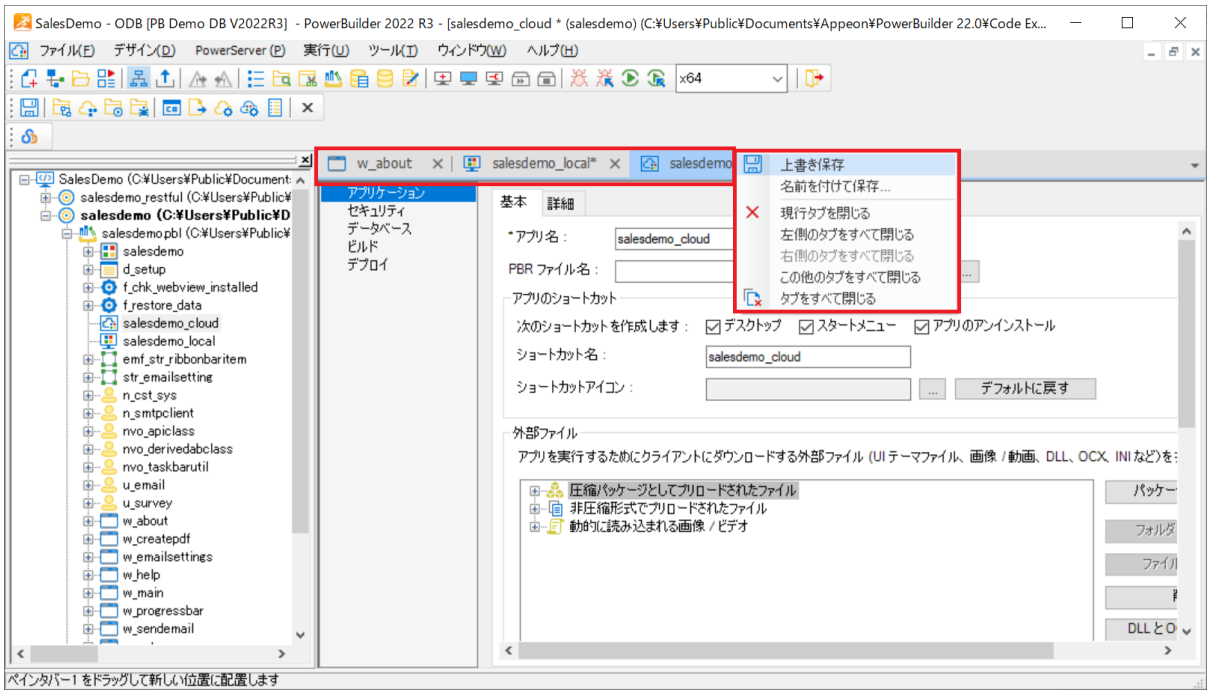
- MDI ではなくタブで複数のエディター / ペインターを開くことをサポートします。

  PB.INI ファイルでは、**NewTabAtRightMost** という設定がサポートされています。この設定は、ペインター / エディターを IDE のタブ バーの左端から開くか右端から開くかを決定します。詳細については、ユーザー ガイドのセクション 9.2.2「PB.INI 設定」を参照してください。

```
[Tabbedbar] NewTabAtRightMost=0 //0: left most location. 1: default value, right most location.
```

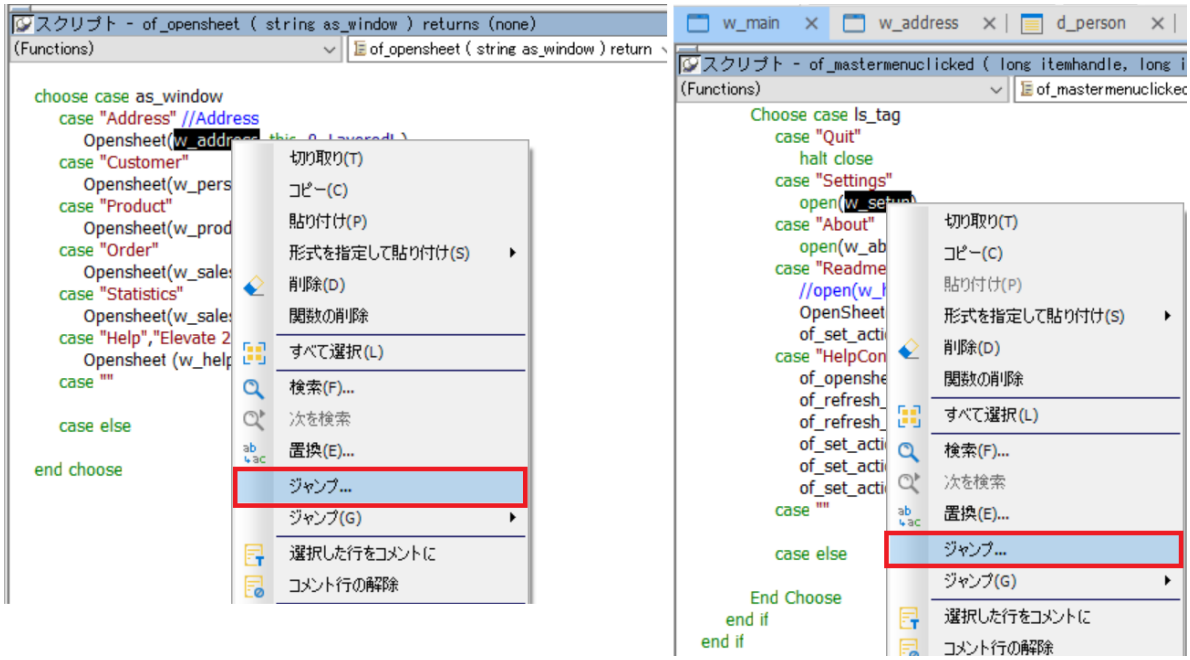
  タブを右クリックすると、ポップアップ メニューから現在のオブジェクトを保存したり、現在のタブを閉じる、左側のタブを閉じる、右側のタブを閉じる、これ以外のタブを閉じる、すべてのタブを閉じるなどのさまざまな方法でタブを閉じることができます。

図 1.2:



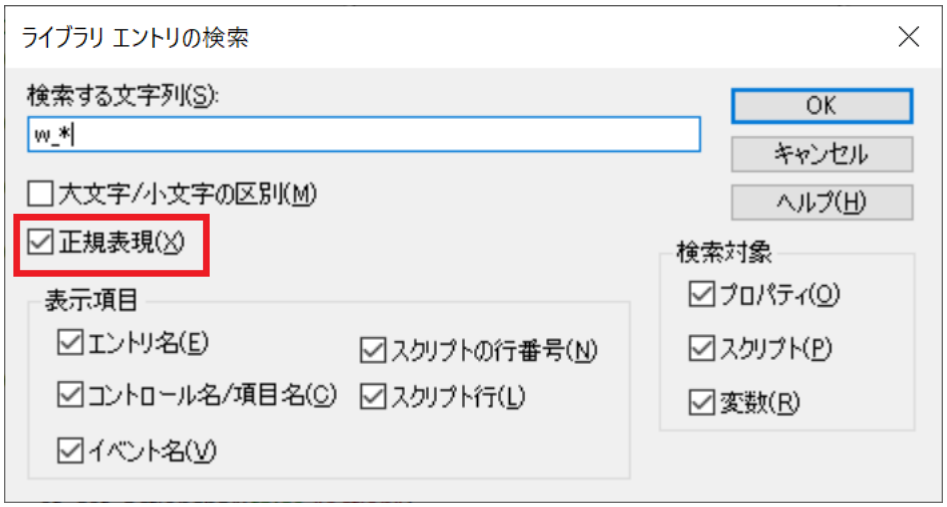
- スクリプト エディターまたはソース編集ウィンドウから選択したオブジェクトへのジャンプをサポートします。ジャンプできるのはオブジェクトのみで、関数、イベント、またはオブジェクト変数にはジャンプできません。ジャンプする前に、まずオブジェクト名を選択する必要があります。同じ名前のオブジェクトが複数ある場合は、PBL リストで最初に見つかったオブジェクトにジャンプします。

Figure 1.3:



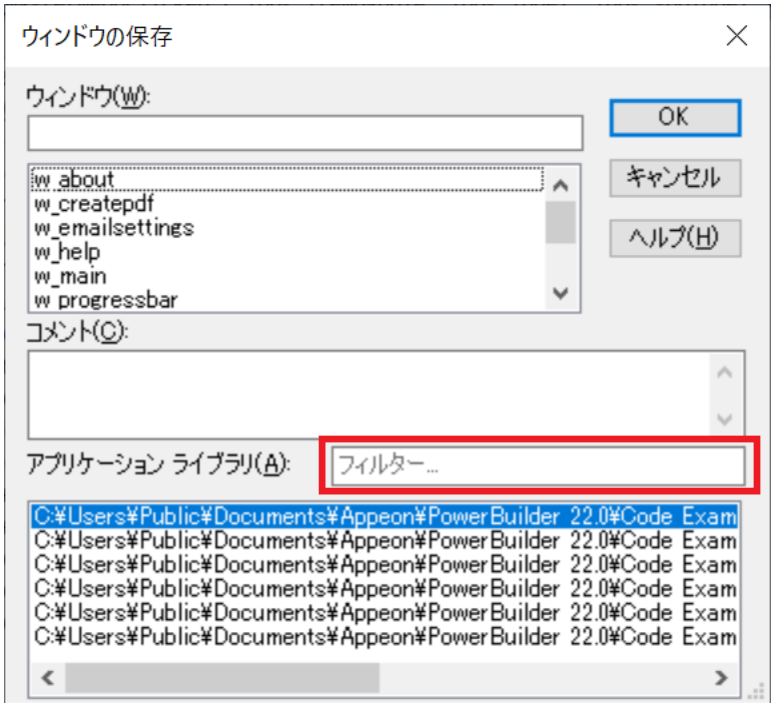
- PBL でも正規表現による検索をサポートします (スクリプトだけでなく)。

図 1.4:



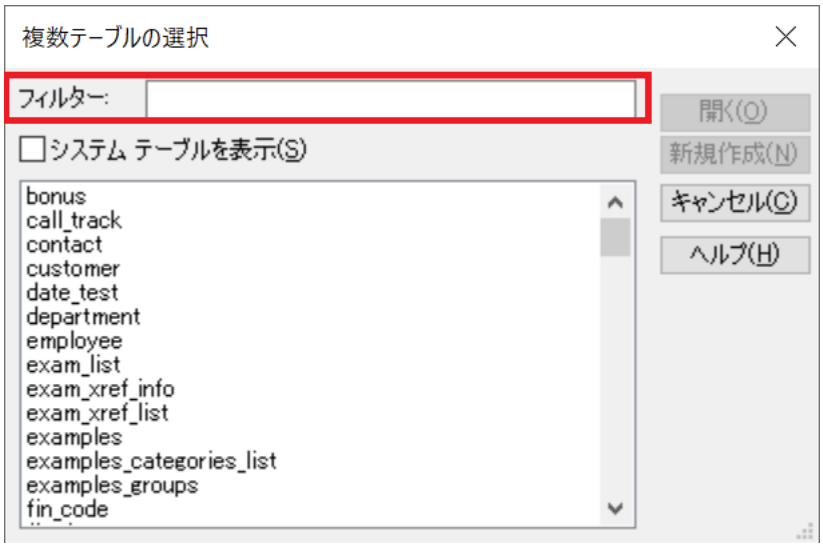
- [保存] ダイアログでアプリケーション ライブラリをフィルターするためのフィルター ボックスを追加します。

図 1.5:



- SQL Select を使用してデータウィンドウを作成するときに選択するテーブルをフィルターするためのフィルター ボックスを追加します。

図 1.6:



- ストアド プロシージャを使用してデータウィンドウを変更するときに選択するストアド プロシージャをフィルターするためのフィルター ボックスを追加します。

1.1.12 IDE で 64-bit アプリケーションを実行またはデバッグする

PowerBuilder IDE は 32-bit ですが、IDE で直接 64-bit アプリケーションを実行またはデバッグできるようになりました。

PowerBuilder IDE は、64-bit アプリケーションの実行またはデバッグに 64-bit デバッグサービスプロセス (PB220DebugService_x64.exe) を使用します：

1. 「32-bit アプリケーションから 64-bit への移行」ドキュメントの移行手順の指示に従って、必要なデータベース接続の変更とアプリケーションのコード変更を行い、アプリケーションを 64-bit として実行できるようにします。

2. ツールバー (パワーバー 1) でビット数オプションを x64 に設定します。このオプションは、IDE でアプリケーションを実行するかデバッグするかに関係なく適用されます。

3. デバッグするプロジェクト / ワークスペース / ターゲットを選択します。デバッグ機能は、32-bit アプリケーションのデバッグ機能と同じです。

   - 従来の C/S アプリケーション プロジェクトをデバッグするには、ツールバーの [デバッグ] (または [選択してデバッグ]) 項目を選択するか、ワークスペースまたはターゲットの右クリック メニューの [デバッグ] 項目を選択します。

   - PowerClient プロジェクトをデバッグするには、PowerClient プロジェクト ツールバーの [デバッグ] 項目を選択します。

   - PowerServer プロジェクトをデバッグするには、固有の要件を満たした後、PowerServer プロジェクト ツールバーの [デバッグ] 項目を選択します。PowerServer プロジェクトから展開されたインストール可能なクラウド アプリでは、クライアント アプリは PowerBuilder IDE のデバッグ オプションに従って実行されますが、PowerServer Web API は C# IDE (つまり、SnapDevelop IDE) で構成されたビット数でコンパイルされることに注意してください。

4. 実行するプロジェクト / ワークスペース / ターゲットを選択します。

- 従来の C/S アプリケーション プロジェクトを実行するには、ツールバーの [実行] (または [選択して実行]) 項目を選択するか、ワークスペースまたはターゲットの右クリック メニューの [実行] 項目を選択します。

- PowerClient プロジェクトを実行するには、PowerClient プロジェクト ツールバーで [PowerClient プロジェクトの実行] 項目を選択します。

- PowerServer プロジェクトを実行するには、固有の要件を満たした後、PowerServer プロジェクト ツールバーの [PowerServer プロジェクトの実行] 項目を選択します。PowerServer プロジェクトから展開されたインストール可能なクラウド アプリでは、クライアント アプリは PowerBuilder IDE のビット数オプションに従って実行されますが、PowerServer Web API は C# IDE (つまり、SnapDevelop IDE) で構成されたビット数でコンパイルされることに注意してください。

1.1.13 インストーラーの機能強化

- 製品名「PowerBuilder Compiler」は、より一般的な名称「PowerBuilder Utilities」に変更されました (これには、OrcaScript ツール (orcascr220.exe)、PowerBuilder Compiler ツール (pbc220.exe)、および PBAutoBuild ツール (PBAutoBuild220.exe) が含まれます)。

- デモ アプリケーションをインストールするためのオプションは複数あります :

- 「デモをインストールしない」を選択することもできます。その場合、デモ アプリケーションとデータベースはインストールされません。

- デモ アプリケーションには SQL Server デモ データベースが付属しているため、デモ データベースとして「SQL Server」を選択できます。これにより、デモ アプリケーションとデータベースのデータベース タイプとして、SQL Server、SQL Anywhere、または PostgreSQL を選択できるようになりました。

- PowerBuilder インストーラーを使用すると、IIS サーバーをローカル Web サーバーとして設定できます。PowerClient / PowerServer プロジェクトのデフォルトサーバー プロファイルとして既存の Web サイトを選択したり、新しい Web サイトを作成したり、後で Web サイトを指定したりすることができます。

- PowerBuilder / InfoMaker インストーラーは、管理者権限を必要とせずに非管理者ユーザー (標準ユーザーまたはゲスト ユーザーとして Windows にログインし、インストーラーをインストールして起動し、オフライン インストーラーを今すぐダウンロードできます) に利便性を提供すると同時に、システムを不正な変更から保護するように強化されています (非管理者ユーザーは、製品とコンポーネントをインストールするために管理者の資格情報を入力する必要があります)。

  要約すると、インストーラーは次の側面から強化されています :

  1. PowerBuilder / InfoMaker インストーラーのインストール (PowerBuilderInstaller_bootstrapper.exe の実行) に管理者権限は不要になりました。
  2. PowerBuilder / InfoMaker インストーラーを起動する場合 (Windows のスタート メニューから、または PowerBuilderInstaller.exe を実行して)、管理者権限は不要になりました。
  3. オフライン インストール パッケージをダウンロードする場合 (インストーラーで [オフライン インストーラーのダウンロード] をクリックする) に管理者権限は不要になりました。

  ただし、システム ファイル、レジストリ エントリ、および重要な構成に変更を加える場合は、管理者権限が必要です。たとえば、インストーラーで [変更]、[更新]、または [インストール] をクリックして製品をインストールまたは更新しようとすると、続行する前に管理者のユーザー名とパスワードを入力するように求められます。

1.1.14 .NET アップグレード

1.1.14.1 .NET 8.0 にアップグレード

次の機能は、.NET 8.0 で動作するようにアップグレードされました :

- コンピューター上で .NET SDK 8.0 が検出されない場合は、PowerBuilder インストーラーによってインストールされます。
- ADO.NET データベース接続には、SQL Server、Oracle、および ODBC 接続用の .NET 8 データ プロバイダーが必要になりました。
- .NET DLL インポーターは、.NET 8.0 DLL をインポートできます。

- .NET DLL インポーターは、フレームワーク タイプ オプションが提供されなくなりました。
- 推奨されるフレームワーク タイプは .NET ですが、.NET Framework または .NET Core DLL をインポートすることも可能です。
- LoadWithDotNetFramework 関数と LoadWithDotNetCore 関数は廃止されました。LoadWithDotNetFramework 関数または LoadWithDotNetCore 関数を呼び出すスクリプトは、LoadWithDotNet 関数を呼び出すスクリプトに置き換える必要があります。
- LoadWithDotNet 関数 (および対応するランタイム ファイル pbdotnetinvoker.dll) は、.NET、.NET Framework、または .NET Core DLL を読み込むことができます。
- 推奨される DLL バージョンは、.NET 6.0 または 8.0 ですが、それより低いバージョンの DLL をインポートすることも可能です。
- PowerBuilder アプリケーションは、DotNetAssembly および DotNetObject を介して .NET 8.0 DLL をロードできます。
- RibbonBar Builder、Image Selector、および .NET DLL Importer ツールは、.NET 8 を使用して構築されています。

1.1.14.2 .NET Framework 4.8 にアップグレード

次の機能は、.NET Framework 4.8 で動作するようにアップグレードされました：

- コンピューター上で .NET Framework 4.8 が検出されない場合、PowerBuilder インストーラーによってそれがインストールされます。
- PowerBuilder ランタイム パッケージャーが動作するには .NET Framework 4.8 が必要です。
- DataWindow BLOB コントロールが動作するには、.NET Framework 4.0 以降の互換性のあるバージョン (4.8 など) が必要です。
- Excel12 のサポートには、.NET Framework 4.0 以降の互換バージョン (4.8 など) が必要です。

1.1.15 ソースコントロールの強化

- (Git のみ) Git ソース コントロール システムから PowerBuilder ワークスペースを取得するときに ([ワークスペースに接続] コンテキスト メニューを使用して)、Windows 資格情報または Github サインイン ページのトークンを使用して Git にサインインできます (Windows 資格情報または Github サインイン ページを表示するには、まず **Git for Windows** ツールをインストールする必要があります) 。

- (Git のみ) Git ソース コントロール システムから PowerBuilder ワークスペースを取得するときに ([ワークスペースに接続] コンテキスト メニューを使用して)、ブランチを取得するように指定できます。ブランチを指定していない場合は、デフォルトで "main" になります。

- (Git および SVN) ソース コントロール システムから PowerBuilder ワークスペースを取得するときに ([ワークスペースに接続] コンテキスト メニューを使用して)、ソース コードから PBL を生成するように選択できます。したがって、PBL をソース コントロールにアップロードする必要はなくなりました。

- (Git および SVN) [ソースコントロールに追加] コンテキスト メニューを使用してワークスペースをアップロードすると、PBL ファイルはファイル / フォルダー リストにアップロード対象として表示されなくなります。必要に応じて、[PBL のアップロード] コンテキスト メニューを使用して PBL ファイルをアップロードできます。ただし、PBL はソース コードから自動的に生成できるため、ソース管理用に PBL をアップロードする必要は通常ありません。

- (SVN) SVN ソースコントールでサポートされている OpenSSL がバージョン 3.1.2 にアップグレードされました。

- (Git および SVN) ライブラリ ペインターでソースコントロールコマンドを実行できます。PB ワークスペースをソースコントロールに追加したら、ライブラリ ペインターを使用してその中のアプリケーション ファイルを開くと、ファイルの右クリック メニューに関連するソースコントロールコマンドが表示されます。

詳細については、ユーザー ガイドの「ソースコントロールの使い方」を参照してください。

1.1.16 マイグレーションアシスタントの最適化

マイグレーション」アシスタントはさまざまな面で最適化されています ：

- UI

  ウィザードの視覚的な表示 (背景画像、ウィンドウ サイズなど) が再設計されました。

- UX

  ウィザード ウィンドウのサイズを変更できるようになりました。

  [検索するライブラリの指定] ウィンドウでは、1) ドラッグ、2) ダブルクリック、3) チェックボックスを選択してドラッグ、4) 右クリックしてポップアップ メニュー (フォルダー内のすべてを選択またはライブラリを選択) を選択するなど、複数の方法でライブラリを検索リストに追加できます。

- 機能

  マイグレーションアシスタントは、以前のバージョンからアプリケーションを移行するときに、より多くの構文をキャプチャしたり、移行の提案を提供したりできます。

表 1.2:

構文	推奨代替品（古い）	推奨代替品 (最適化済み)
FromUnicode	Unicode 文字列を含む BLOB を変換する廃止された関数	String(blob)
ToUnicode	文字列を Unicode BLOB に変換する廃止された関数	Blob(string)
FromAnsi	廃止された ANSI 関数	String(blob,EncodingANSI!)
ToAnsi	廃止された ANSI 関数	Blob(string,EncodingANSI!)
Excel!	廃止された SaveAsType データウィンドウ定数	Excel8!、XLSB!、または XLSX!
WMF!	廃止された SaveAsType データウィンドウ定数	EMF!
Inet	inet オブジェクトの廃止	OpenURL 関数の仕様に置き換え

AddressChange	利用不可	AddressChange (string newurl, boolean isNewPage)
PdfPrintFinished	利用不可	PdfPrintFinished (string pdfFile, boolean result, integer errorcode)
DownloadingStart	利用不可	DownloadingStart(integer itemId, string fileName, string uri, string mimeType, string contentDisposition, longlong total)
構文	**推奨代替品（古い）**	**推奨代替品 (最適化済み)**
EvaluateJavascriptFinished	利用不可	EvaluateJavascriptFinished(bon isFinish, string result, string errorMessage)
NavigationStart	利用不可	NavigationStart(boolean isRedirected, string requestHeaders, string uri)
LoadWithDotNetFramework	利用不可	LoadWithDotNet (readonly string assemblypath)
LoadWithDotNetCore	利用不可	LoadWithDotNet (readonly string assemblypath)

1.1.17 Oracle の ID 列をサポート

Oracle データベースの ID 列をサポートします：

- DataWindow オブジェクトが作成されると、Identity 列が自動的に設定されます (データベースから読み取られます)。[更新プロパティの指定] で Identity 列を変更できます。

- Identity 列を含む行を挿入する場合、Identity 列は INSERT ステートメントから除外されます。その値は、GENERATED BY DEFAULT に基づいて生成されます。

- 新しい行が挿入されると、更新後にその行の ID 列の値が表示されます。

  注意：データウィンドウは、更新後に新しく追加された行の Identity 列の最大値を自動的に取得します。したがって、Identity 列が負の数で自動的に増分される場合 (これは Oracle および ADO.NET ではサポートされていますが、ODBC ではサポートされていません)、新しく追加された行に表示される値は正しくない可能性があります。正しい値を取得するには、データウィンドウで Retrieve を実行できます。

- IDE データベース ペインタで Oracle テーブルの自動増分列を直接追加できるようになりました。列のデータ型を NUMBER IDENTITY または FLOAT IDENTITY に設定することで実行できます。句は次のようになります:

```
GENERATED ALWAYS AS IDENTITY INCREMENT BY 1 START WITH 1
```

1.1.18 SQL Server の "Strict" 暗号化をサポート

DB プロファイルの [データの暗号化] オプションに、新しい暗号化タイプ "Strict" が追加されました。 "Strict" 暗号化タイプにより、SQL Server 2022 で TDS 8.0 を活用できるようになります。TDS 8.0 と Strict 暗号化の詳細については、https://learn.microsoft.com/en-us/sql/relationaldatabases/security/networking/tds-8?view=sql-se rver-ver16 を参照してください。

"Strict" は、次の SQL Server ドライバーを選択した場合に使用できます：

- SQL Server 用 OLE DB ドライバー (MSOLEDBSQL 19.x)

```
SQLCA.DBMS = "MSOLEDBSQL SQL Server"
SQLCA.AutoCommit = False
SQLCA.DBParm = "Provider='MSOLEDBSQL19',Encrypt=2"
```

- SQL Server 用 ADO.NET プロバイダー

```
SQLCA.DBMS = "ADO.Net"
SQLCA.AutoCommit = False
SQLCA.DBParm = "Provider='SQL Server',PROVIDERSTRING='Encrypt=Strict;'"
```

- MSOLEDBSQL 18.x は "Strict" 型をサポートしていません。
- SNC SQL ネイティブ クライアントは "Strict" タイプをサポートしていません。

詳細については、セクション 1.1.52 「暗号化」を参照してください。

PowerServer では厳密な暗号化もサポートされています。詳細については、こちらを参照ください。

1.1.19 ADO.NET ドライバーのアップグレード

ADO.NET データベース ドライバーのアップグレードとリファクタリングにより、データベースの基盤となるドライバーが変更されました。

現在、基盤となるデータベース ドライバーは、Oracle、SQL Server、および ODBC 経由の接続のみをサポートしています (ODBC の場合、IDE は ASE と SQL Anywhere のみをサポートしています)。 詳細については、「データベースへの接続」の「*ADO.NET インターフェイスの使用*」を参照してください。

データベース プロファイル設定、データベース トレース、および PowerBuilder ランタイム パッケージャー ウィンドウは、変更に応じて調整されました：

- ADORelease、DataLink、DbConfigSection、Namespace などの一部のデータベース パラメーターはサポートされなくなりました (データベース プロファイル設定ウィンドウのオプションは調整されています)。
- データベース トレース ファイルには DBI コマンドの名前が含まれなくなります (そのため、データベース トレース ウィンドウから **[DBI 名の表示]** オプションが削除されます)。
- 選択されたデータベース接続に応じて、PowerBuilder ランタイム パッケージャーによってさまざまなランタイム ファイルがパッケージ化されます。
- Sybase.PowerBuilder.DataSource.Sharing.dll (ADO.NET データベース接続の共有用) がアップグレードされ、**Appeon.DB.Sharing.dll** に名前が変更されました。

1.1.20 モダン グラフ

グラフの全体的な表示は、次の点から改善されました (比較画像はこちらで確認できます)：

- 元グラフの 14 色の古い色は、現在の主流のスタイルによる新しい配色に置き換えられました。
- 新しいスタイルのグラフは、よりフラットでモダンなデザインになっているだけでなく、アップグレードされたテクノロジー GDI+ を利用して、線をより滑らかにしギザギザのエッジを排除しています。
- 3D スタイルのグラフは新しい配色のみをサポートしますが、線のスタイルは従来のスタイルのままです。
- 2D スタイルの棒グラフとヒストグラム値軸の実線と凡例の境界線が削除されました。
- 折れ線グラフのデフォルトの白抜きボックス シンボル値は固定の白抜きドットに置き換えられ、シンボル値の変更はサポートされなくなりました。
- 散布図の値が実線のドットに置き換えられました。
- モダン グラフ スタイルとテーマの両方を適用すると、テーマの配色が優先され、2D スタイル グラフの線スタイルにはモダン スタイルが使用されます。

pb.ini ファイルで最新のグラフスタイルを有効にすることができます：

```
[Application]
ModernGraph=1
```

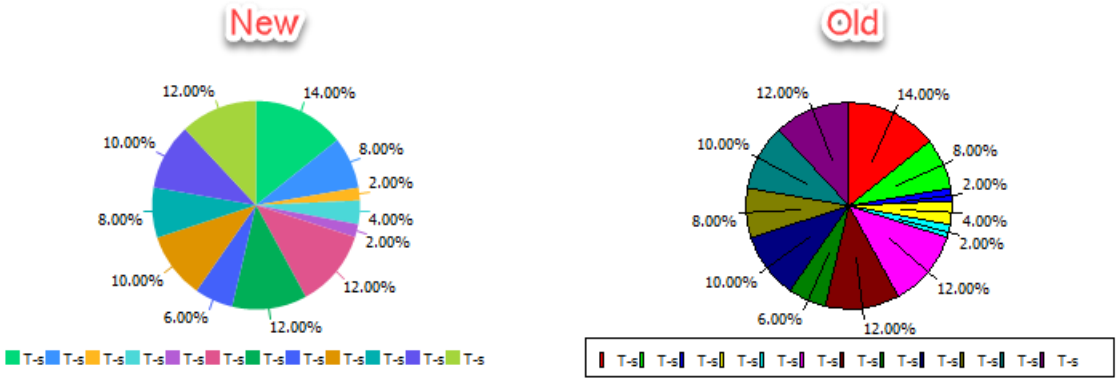
"1" はモダン スタイル、**"0"** はオリジナルの伝統的なスタイルを表し、デフォルト値は **"1"** です。

1.1.21 モダンなグラフと伝統的なグラフ

グラフの全体的な表示は、以下の点から改善されました：

- 元のグラフの 14 色の古い色は、現在の主流のスタイルに従った新しい配色に置き換えられました。

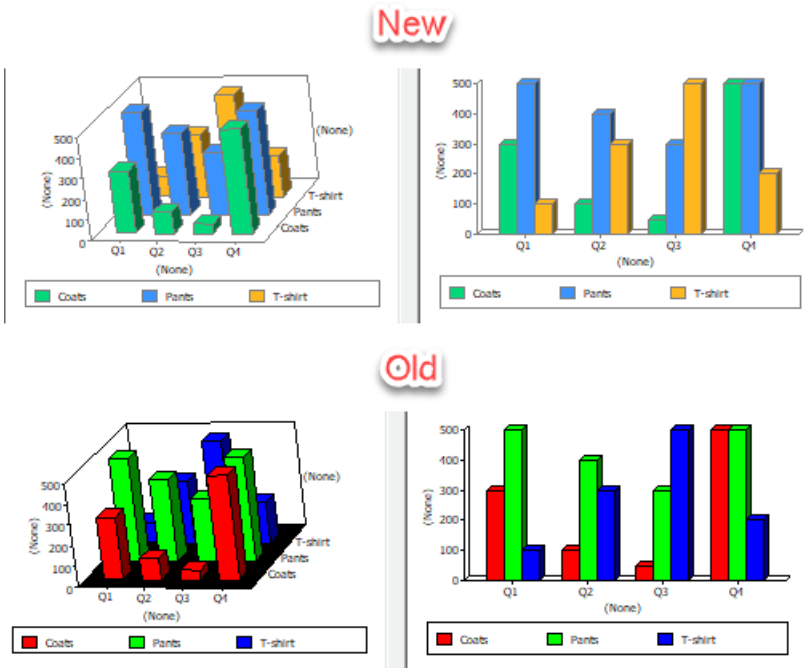
比較 1.7：



- スタイルのグラフは、よりフラットでモダンなデザインになっただけでなく、アップグレードされたテクノロジー GDI+ を利用して、線をより滑らかにし、ギザギザのエッジを排除しています。

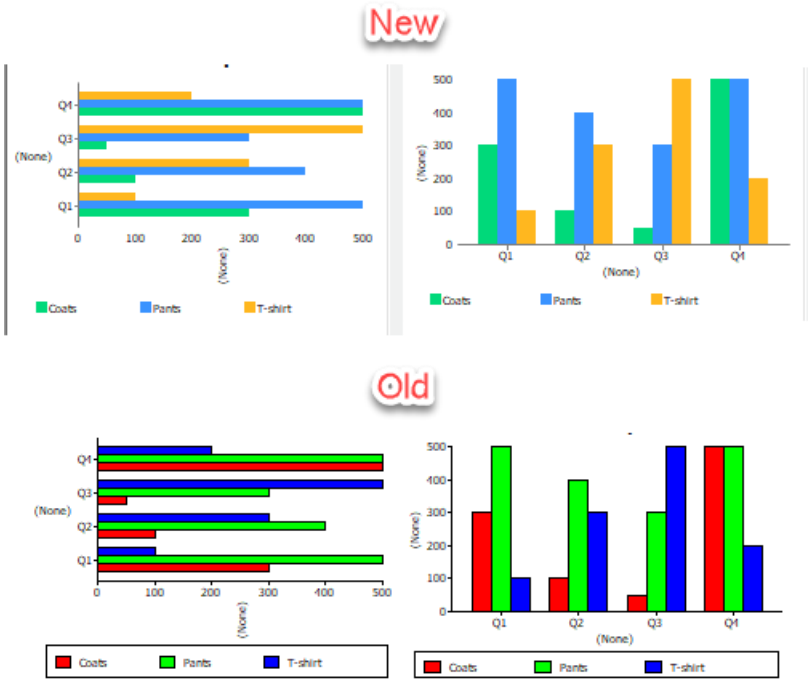
- 3D スタイルのグラフは新しい配色のみをサポートしますが、線のスタイルは元の従来のスタイルのままです。

比較 1.8：



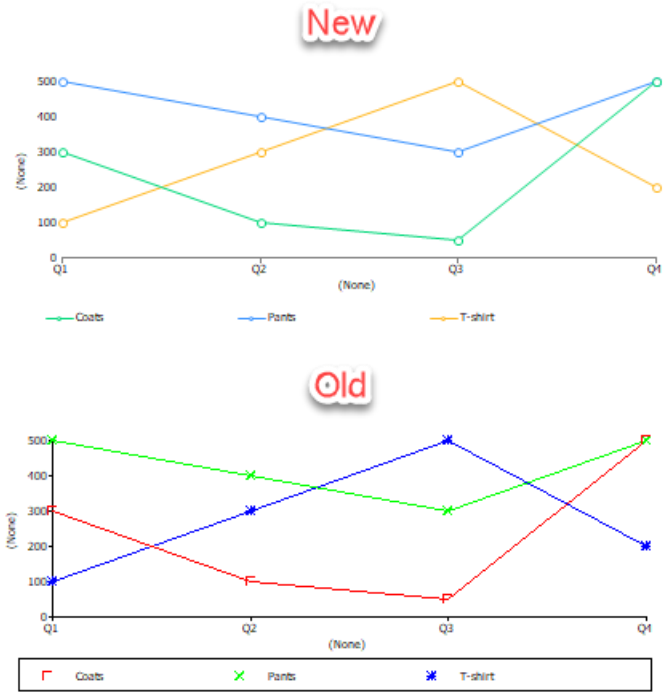
- 2D スタイルの棒グラフとヒストグラム値軸の実線と凡例の境界線が削除されました。

比較 1.9：



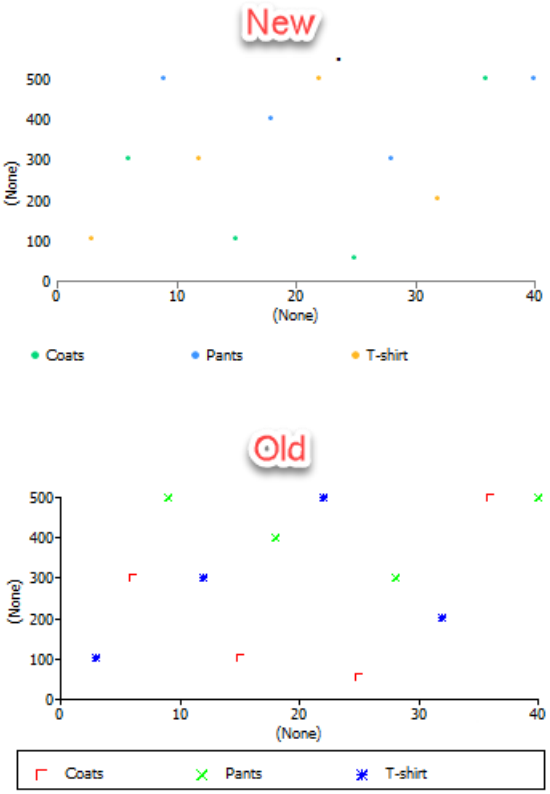
- 折れ線グラフのデフォルトの白抜きボックス シンボル値は固定の白抜きドットに置き換えられ、シンボル値の変更はサポートされなくなりました。

比較 1.10：



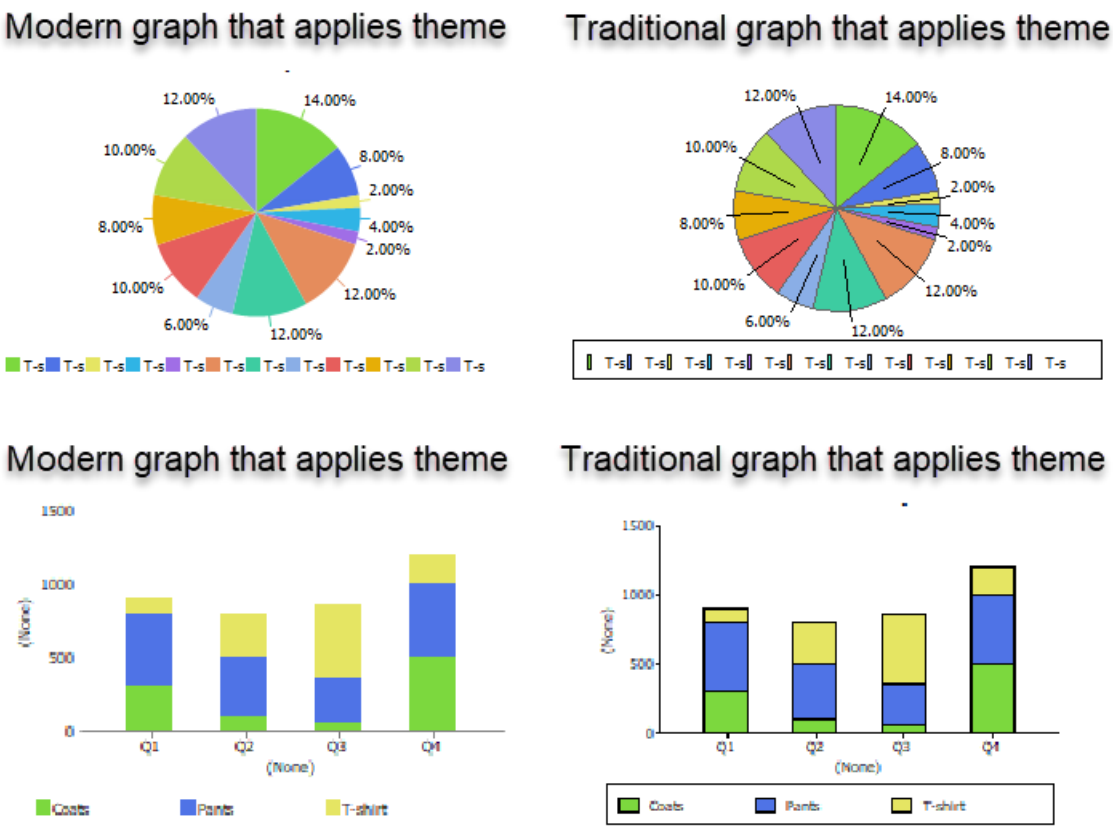
- 散布図の値が実線のドットに置き換えられました。

比較 1.11:



- モダングラフスタイルとテーマの両方が適用されている場合、テーマの配色が優先されますが、2D スタイルグラフの線スタイルにはモダンスタイルが使用されます。

比較 1.12:



1.1.22 アクセシビリティの改善

PB.INI のアクセシビリティ オプションを使用すると、(DataWindow だけでなく) すべてのコントロールのアクセシビリティ機能をオフにできるようになりました。

以前のバージョンでは、PB.INI ファイルで accessibility=0 を指定すると、DataWindow コントロールのアクセシビリティのみがオフになり、他のコントロールはオフになりませんでした。その結果、実行時には常に **pbacc.dll** と **PBAccessibility.dll** が必要になりました

2022 R3 では、accessibility=0 の場合、すべてのコントロールのアクセシビリティがオフになります。アクセシビリティ オプションは **[Data Window]** セクションから **[Application]** セクションに移動され、PowerBuilder ランタイム パッケージャーに新しいオプションの **Accessibility Support** が追加され、**Accessibility Support** オプションが選択されている場合にのみ **pbacc.dll** と **PBAccessibility.dll** が必要になります。

1.1.23 デモアプリの変更

1.1.23.1 ウェブサイトからデモを実行してダウンロードする

デモ アプリケーション (PowerBuilder、PowerServer、および .NET DataStore 用) は、専用の Web サイト https://demo.appeon.com/ から直接実行できます (または、PowerBuilder IDE から [ヘルプ] | [PowerBuilder デモ アプリ] メニューをクリックして Web サイトにアクセスします) 。

PowerBuilder のインストール プロセス中に PowerBuilder デモ アプリケーションとデータベースがインストールされていない場合 (たとえば、インストール中に **PowerBuilder チュートリアル**、**デモ データベース**、および**コードサンプル**のコンポーネントの選択を解除した場合)、デモ アプリケーションのソース コード (サンプル アプリ、グラフ サンプル アプリ、セールス サンプル アプリ) とデータベース ファイル (SQL Anywhere データベースまたは PostgreSQL データベース) を手動でダウンロードし、ODBC データ ソースを自分で設定して PowerBuilder デモ アプリケーションを実行できます。詳細については、インストール ガイドのセクション 4.3 「PowerBuilder デモ アプリケーションとデータベースのインストール」を参照してください。

1.1.23.2 .NET アセンブリ呼び出し用の新しいデモ アプリ

.NET アセンブリ呼び出しを紹介する新しいデモ アプリケーションが追加されました。このデモには、Windows のスタート | Appeon PowerBuilder 2022 R3 | Example Components App からアクセスできます。

このサンプルコンポーネントアプリでは、以下のサンプルを説明しています：

- .NET 関数との間でのデータの送受信、.NET クラスの処理とアクセス、C# コードからの PB コールバックの実行などの基本的な使用方法について
- QR コード、バーコード、FTP、XML エディター、DDE、OpenAI、OS 情報の取得などの一般的なシナリオについて

1.1.24 利用可能なドキュメント

PB.INI ファイルの設定に関するドキュメントは、以下の場所で入手できます ：
https://docs.appeon.com/ pb2022r3/pbug/PBINI_settings.html

PowerBuilder のトラブルシューティングのヒントに関するドキュメントは、以下の場所で入手できます：https:// docs.appeon.com/pb2022r3/troubleshooting_guide/

データウィンドウの PushCutControlToNextPage プロパティが完全にサポートされるようになりました (データウィンドウの印刷、印刷プレビュー、NativePDF、および Distill)。PushCutControlToNextPage プロパティは、ネストされたレポートまたは複合データウィンドウで、改ページがあると列のテキストが不適切に分割される可能性がある問題を解決するために使用されます。たとえば、1 行の文字が垂直に 2 つの部分に分割され、1 行目には 1 ページの文字の上部が表示され、もう 1 行目には次のページの文字の下部が表示されます。この場合、PushCutControlToNextPage=Yes を設定すると、列全体を次のページに移動して表示できます。詳細なドキュメントについては、PushCutControlToNextPage プロパティを参照してください。

1.1.25 機能の削除と実行時の違い

PowerBuilder 2022 R3 ではランタイム ファイルが最適化され、TE Edit Control、Java、JDK 1.6、EasySoap、SOAP クライアント、OData などの機能が正式に削除されました。PowerBuilder アプリケーションをアップグレードする前に、このような機能の削除とランタイムの違いの影響を認識しておく必要があります。

たとえば、

実行時の違いについては、「実行時の違い」を参照してください。

PBDOM の変更については、「PBDOM の変更」を参照してください。

Java サポートと JDK 1.6 の削除については、「Java サポートと JDK 1.6 の削除」を参照してください。

アプリのアップグレードに影響する可能性のあるその他のすべての変更の詳細については、「PowerBuilder アプリケーションのアップグレード」を参照してください。

1.2 廃止された機能

PowerBuilder には新しい機能が追加されていますが、既存の機能の一部は不要になったり、使用が推奨されなくなったりする場合があります。

廃止済機能とは、製品から完全に削除された機能です。

廃止された機能は引き続き使用できますが、テクニカル サポートの対象外となり、機能強化も行われなくなります。

これらの**機能の移行パス**については、製品可用性マトリック Web ページの廃止済 / 廃止された機能を参照してください。

以下は、バージョン 2017 R3 以降に廃止 / 廃止されたと宣言された機能の完全なリストです。

表 1.3:

バージョン	タイプ	機能
2022 R3	廃止	DDE 関数とイベント
	廃止	LoadWithDotNetFramework および LoadWithDotNetCore 関数。これらの関数は LoadWithDotNet 関数に置き換えることができます。
	廃止済	Windows 8.1 および Windows Server 2012 R2 OS
	廃止済	DirectConnect データベース接続。ODBC またはネイティブ データベース接続の使用に切り替えることができます。
	廃止済	組み込みのリッチ エディット コントロール (TE Edit Control)。TX テキスト コントロールに置き換えることができます。
	廃止済	Java と JDK 1.6 サポート。回避策については、appeon サポート (support@appeon.com) にお問い合わせください。
	廃止済	Web サービス データウィンドウ w
	廃止済	OData データ ソース e
	廃止済	.NET Web サービス ターゲット (SOAP 使用)
	廃止済	SOAP サーバーに接続するための Web サービス プロ式を作成します。 SOAP サーバーに接続するための Web サービス プロキシを EasySoap および SOAP クライアントの代わりに、HTTPClient オブジェクトを使用して SOAP Web サービス services を呼び出すことができます。
	廃止済	Windows 7 OS
	廃止済	.NET Assembly ターゲット

	廃止	PBC (PowerBuilde コンパイラー)。代わりに PBAutoBuild を使用できます。
	廃止	Microsoft Active Accessibility (MSAA)
2019 R3	廃止	組み込みのリッチ エディット コントロール (TE Edit Control)
	廃止	Inet オブジェクト (GetURL、PostURL、HyperlinkToURL)
	廃止	OLEControl で IE ブラウザーページを開く
	廃止	Windows 7 OS
	廃止	ドッキングウィンドウ
	廃止	Windows Server 2008 OS
バージョン	**タイプ**	**機能**
	廃止	.NET Web サービス ターゲット (SOAP を使用)
	廃止	.NET アセンブリ ターゲット
2017 R3	廃止	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)
	廃止	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
	廃止済	Web データウィンドウ
	廃止	EJB クライアント

1.3 バグ修正と既知の問題

https://docs.appeon.com/pb2022r3jp/release_bulletin_for_pb/ を参照してください。