Objects and Controls

Appeon PowerBuilder®

2017

DOCUMENT ID: DC37787-01-1700-01

LAST REVISED: July 2017

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

Contents

About This Book

Audience

This book is for programmers who use PowerBuilder® to build client/server or multitier applications. It describes the system-defined objects in PowerBuilder and their default properties, functions, and events.

Related documents

For detailed information about the properties, functions, and events described in this book, see the *PowerScript Reference* and the *DataWindow Reference*. For a complete list of PowerBuilder documentation, see *PowerBuilder Getting Started*.

Other sources of information

Use the Appeon Product Manuals web site to learn more about your product. The Appeon Product Manuals web site is accessible using a standard Web browser.

To access the Appeon Product Manuals web site, go to Product Manuals at https://www.appeon.com/developers/library/product-manuals-for-pb.

The installation guide in PDF format can be accessed from the PowerBuilder installation package. The release bulletin can be access from Online Help at https://www.appeon.com/support/documents/appeon_online_help/pb2017/release_bulletin_for_pb.

Conventions

The formatting conventions used in this manual are:

Formatting example	Indicates
Retrieve and Update	When used in descriptive text, this font indicates: • Command, function, and method names • Keywords such as true, false, and null • Datatypes such as integer and char • Database column names such as emp_id and f_name • User-defined objects such as dw_emp or w_main

Formatting example	Indicates
variable or *file name*	When used in descriptive text and syntax descriptions, oblique font indicates: • Variables, such as *myCounter* • Parts of input text that must be substituted, such as *pblname*.pbd • File and path names
File>Save	Menu names and menu items are displayed in plain text. The greater than symbol (>) shows you how to navigate menu selections. For example, File>Save indicates "select Save from the File menu."
`dw_1.Update()`	Monospace font indicates: • Information that you enter in a dialog box or on a command line • Sample script fragments • Sample output fragments

If you need help

All customers are entitled to standard technical support for reproducible software defects. You can open a standard support ticket at the Appeon support site: https://www.appeon.com/standardsupport/ (login required).

If your organization has purchased a premium support contract for this product, then the designated authorized support contact(s) may seek assistance with your technical issue or question at the Appeon support site: https://support.appeon.com (login required).

CHAPTER 1

PowerBuilder System Objects and Controls

About this chapter

This chapter provides overview information about PowerBuilder system objects and controls. This chapter also lists the PowerBuilder system objects not included in this book and explains why they are not included.

Contents

What are system objects?

System objects

PowerBuilder **system class object**s are the built-in objects you use to develop your application. PowerBuilder system objects include objects such as windows and menus, as well as graphical controls and predefined entities that you can reference in your application, such as the Message and Error objects.

Controls

PowerBuilder **control**s are a subset of system objects that you place in windows or user objects. Typically, they are graphical objects that allow users to interact with your application or that you use to enhance the design of your windows.

System structures

PowerBuilder system **structure**s are a subset of system objects that contain properties that describe the state of other system objects or the system itself. For example, the Environment object is a structure that holds information about the computing platform the PowerBuilder Application object is running on.

System object properties, events, and functions

Properties

Each system object has a number of **properties** associated with it that define its characteristics. For example, the CheckBox control has Height and Width properties that control its size and a BackColor property that controls its background color. You can set the value of object properties within scripts or with the object's Property sheets available within the painters.

Events

PowerBuilder applications are **event**-driven. For example, when a user clicks a button, chooses an item from a menu, or enters data into an edit box, an event is triggered. You write scripts using PowerScript®, the PowerBuilder language, that specify the processing that should happen when the event is triggered. PowerBuilder passes arguments to events, such as the coordinates of the pointer, that help your application figure out what the user did to trigger the event. For most events, you can specify a return code to affect what happens next, such as triggering another event.

Controls, with the exception of the GroupBox and the drawing objects (Line, Oval, Rectangle, and RoundRectangle), always have events related to them. Some system objects, such as system structures, have no events associated with them.

Functions

PowerScript provides a rich assortment of built-in **function**s you can use to act upon the objects and controls in your application. For each system object, there is a set of these built-in functions that can act on it. You use these functions in scripts to manipulate the object.

System object inheritance hierarchy

Inheritance

One of the most powerful features of PowerBuilder is **inheritance**. It enables you to build windows, user objects, and menus that are derived from existing objects. When you build an object that inherits from another object, you create a hierarchy (or tree structure) of ancestor and descendent objects.

Base class object

The object at the top of the hierarchy is a **base class object**, and the other objects are descendants of this object. Each descendant inherits its definition from its ancestor. The base class object typically implements generalized processing, and each descendant modifies the inherited processing as needed.

System object hierarchy

The PowerBuilder system objects compose such a hierarchy. At the top of the hierarchy is the **PowerObject**, the base class from which all the objects and controls described in this book descend. The hierarchy also contains other (generic) base class objects that are not typically used in application development but are necessary parts of the logical organization of the hierarchy.

Undocumented base class objects

Base class objects whose primary function is to provide generic properties and functions for descendent objects are not documented, since these objects typically are not used in applications. The base class system objects that are not documented are:

ClassDefinitionObject
ConnectObject
CPlusPlus
DragObject
DrawObject
DWObject
ExtObject
Function_Object
GraphicObject
NonVisualObject
OmControl
OmCustomControl
OmEmbeddedControl
OmObject
OmStorage
OmStream
ORB
PBtoCPPObject
PowerObject
RemoteObject
Service
Structure
WindowObject

Viewing system objects

Using the Browser

From within PowerBuilder, you can use the PowerBuilder Browser to see a complete list of system objects and their properties, events, and functions.

- To display the system objects, select the System tab of the Browser. The default display is to list the objects alphabetically.
- To see the objects displayed hierarchically, place the cursor in the left pane, press the right mouse button, and select Show Hierarchy.
- To display a specific object's properties, events, or functions, select the object in the left pane and then double-click the Properties, Events, or Functions item in the right pane.

For information about using the PowerBuilder Browser, see the PowerBuilder *Users Guide*.

Using online Help

You can also use PowerBuilder online Help to view more descriptive Help topics about the properties, events, and functions for system objects and controls.

- If you know the name of the system object or control, use the Index tab to go directly to the correct topic.
- To see a list of the system objects and controls for which Help topics exist, select Objects and Controls from the Help contents list.
- In the Browser, select Help from the pop-up menu for the system object or control or one of its functions.

Help not available for base objects
If you select Help from the pop-up menu for a base object that descends from NonVisualObject, the Help topic for NonVisualObject displays. For other base objects, the Help topic for the Browser displays.

CHAPTER 2

System Object Properties, Events, and Functions

About this chapter

This chapter lists the properties, events, and functions of PowerBuilder system objects and controls. This chapter does not include base class objects.

Contents

The objects and controls are listed alphabetically.

ADOResultSet object

The ADOResultSet object provides the ability to use ActiveX Data Object (ADO) record sets to return a result set to a client and to manipulate ADO Recordsets in PowerBuilder.

Properties

ADOResultSet property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

ADOResultSet event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

ADOResultSet function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
GetRecordSet	Integer	Returns the current ADO Recordset
PostEvent	Boolean	Adds an event to the end of the message queue for the object
SetRecordSet	Integer	Sets up the ADOResultSet object to get data from the passed ADO Recordset
SetResultSet	Integer	Populates a new ADOResultSet object with data from the passed ResultSet object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object

Animation control

Animation controls can display Audio-Video Interleaved (AVI) clips that come from an uncompressed AVI file or from an AVI file compressed using run-length encoding (BI_RLE8).

Properties

Animation property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AnimationName	String	Specifies the name of the file that contains the AVI clip. The AVI clip cannot have a sound channel.
AutoPlay	Boolean	Specifies whether the animation starts as soon as the AVI clip is opened. Values are: TRUE – Control plays automatically when opened. FALSE – Control does not play automatically when opened (default).
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Animation property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to put the control into Drag mode manually by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to use to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
OriginalSize	Boolean	Specifies whether the width and height properties of an animation control are set to the size of the AVI clip. Values are: TRUE – Width and height set to original values. FALSE – Existing width and height not changed. In the Window painter, setting OriginalSize to true overrides the existing width and height.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTipText	Long	Specifies a PowerTip for the control.
TabOrder	Integer	Specifies the tab value of the animation within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Transparent	Boolean	Specifies whether the background of the control matches the background of the window it is on, creating a transparent effect. Values are: TRUE – Control is transparent. FALSE – Control is not transparent.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.

Animation property	Datatype	Description
X	Integer	Specifies the X position (distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (distance from the top of the window), in PowerBuilder units.

Events

Animation event	Occurs
Clicked	When the control is clicked (selected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control
Start	When an animation has started playing
Stop	When an animation has stopped playing

Functions

Animation function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible

Animation function	Datatype returned	Description
Move	Integer	Moves the control to a specified location
Play	Integer	Starts playing the AVI clip in the control
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Resize	Integer	Changes the size of the control
Seek	Integer	Displays a specified frame in an AVI clip
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
Stop	Integer	Stops playing the AVI clip in the control
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

Application object

An application is a collection of PowerBuilder windows and objects that provide functionality for user activities, such as order entry or accounting activities. The Application object is the entry point into the applications.

When a user runs an application, the Open event of the Application object is fired. The Open event triggers the script that initiates all the activity in the application.

Properties

Application property	Datatype	Description
AppName	String	Specifies the name of the Application object.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DDETimeOut	Integer	Specifies the number of seconds PowerBuilder acting as the DDE client waits before giving up when trying to communicate with a server using DDE (the default is 20 seconds).
DisplayName	String	User-readable name for your application. This name is displayed, for example, in OLE dialog boxes that show the application's name. If you do not specify a value, the value of AppName is used for DisplayName.
DWMessageTitle	String	Specifies the title of the message box for any runtime DataWindow® errors encountered in the application. If you change the value of this property in script, the new value is recognized only for DataWindows created (or painted) after the new value is set.
FreeDBLibraries	Boolean	Specifies whether you want PowerBuilder to free database interface libraries upon disconnecting from the database. The default is FALSE (PowerBuilder does not free the libraries upon disconnecting).
MicroHelpDefault	String	Specifies the default text of the MicroHelp object (the MicroHelp text that displays when you initiate a PowerBuilder session). The default is Ready.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order in MessageBoxes displayed when you call the MessageBox function. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Message box text displays in right-to-left order. The text of the MessageBox buttons displays in the language of the RightToLeft version of Windows (Arabic or Hebrew) only if you are running a localized version of PowerBuilder. Otherwise, the text of the MessageBox buttons displays in English. FALSE – Characters display in left-to-right order.
ToolbarFrameTitle	String	Specifies the text that displays as the title for the FrameBar when it is floating.
ToolbarPopMenuText	String	Allows you to change the toolbar location text (Left, Top, Right, Bottom, Floating) in the Application's toolbar pop-up menu. Specify the text as a comma-separated list of items.

Application property	Datatype	Description
ToolbarSheetTitle	String	Specifies the text that displays as the title for the SheetBar when it is floating.
ToolbarText	Boolean	Specifies whether the text associated with the items in the toolbar displays. Values are: TRUE – Text displays in toolbar. FALSE – Text does not display in toolbar.
ToolbarTips	Boolean	Specifies whether PowerTips display when text is not displayed on the buttons. Values are: TRUE – PowerTips are displayed. FALSE – PowerTips are not displayed.
ToolbarUserControl	Boolean	Specifies whether users can use the toolbar pop-up menu to hide or show the toolbars, move toolbars, or show text. Values are: TRUE – Users can use pop-up menu. FALSE – Users cannot use pop-up menu.

Events

Application event	Occurs
Close	When the user closes the application.
Idle	When the Idle function has been called in an Application object script and the specified number of seconds have elapsed with no mouse or keyboard activity.
Open	When the user runs the application.
SystemError	When a serious execution time error occurs (such as trying to open a nonexistent application). If there is no script for this event, PowerBuilder displays a message box with the PowerBuilder error number and error message text. For information about error messages, see the *Users Guide*.

Functions

Application function	Datatype returned	Description
ClassName	String	Returns the class of the Application object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.

Application function	Datatype returned	Description
PostEvent	Boolean	Adds an event to the end of the message queue for the Application object.
SetLibraryList	Integer	Sets the PBD library list in an executable. This function can still be used but should be replaced by the system function SetLibraryList.
SetTransPool	Integer	Sets up a pool of database transactions for an application. SetTransPool allows you to minimize the overhead associated with database connections and also limit the total number of database connections permitted.
TriggerEvent	Integer	Triggers a specified event in the Application object and executes the script for the event.
TypeOf	Object	Returns the type of the Application object.

ArrayBounds object

A structure that specifies the upper and lower bounds of a single dimension of an array. It is used in the VariableCardinalityDefinition object. ArrayBounds has no events.

Properties

ArrayBounds property	Datatype	Description
ClassDefinition	PowerObject	Contains information about the class definition of the object or control.
LowerBound	Long	The lower bound of the array dimension. For unbounded arrays, the value is always 0.
UpperBound	Long	The upper bound of the array dimension. For unbounded arrays, the value is always 0.

Functions

ArrayBounds function	Datatype returned	Description
ClassName	String	Returns the class of the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

CheckBox control

CheckBox controls are small square boxes used to set independent options. When they are selected, they display a mark (typically, either an X or a check mark). When they are not selected, they are empty.

Since check boxes are independent of each other, you can group them without affecting their behavior. Grouping check boxes makes the window easier for the user to understand and use.

Typically, check boxes have two states: on and off. You can also use a third state, unknown or unspecified. In the third state, the check is grayed.

Properties

CheckBox property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Automatic	Boolean	Specifies whether the control displays a mark when the user clicks it. Values are: TRUE – Displays mark when clicked. FALSE – Does not display mark when clicked.

CheckBox property	Datatype	Description
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleLowered! StyleRaised!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Move to the top. FALSE – Do not move to the top.
Checked	Boolean	Specifies whether the control is selected. Values are: TRUE – Control is selected. FALSE – Control is not selected.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag Mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag Mode. FALSE – When the control is clicked, the control is not automatically in Drag Mode. You have to put the control into Drag Mode manually by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be clicked). Values are: TRUE – Control can be clicked. FALSE – Control cannot be clicked.
FaceName	String	Specifies the name of the typeface in which the text of the control displays; for example, arial or courier.

CheckBox property	Datatype	Description
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
LeftText	Boolean	Specifies whether the text displays on the left of the control. Values are: TRUE – Text displays on left. FALSE – Text displays on right. Typically, you set this property to false so the text appears on the right of the control.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays next to the control.

CheckBox property	Datatype	Description
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
ThirdState	Boolean	Specifies whether the control is in the third state when the control has been defined to have three states. Values are: TRUE – Control is in third state. FALSE – Control is not in third state.
ThreeState	Boolean	Specifies whether the control has three states. Typically, CheckBox controls have only two states, such as on and off. Values are: TRUE – Control has three states. FALSE – Control does not have three states.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

CheckBox event	Occurs
Clicked	When the control is clicked (selected or unselected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control

CheckBox event	Occurs
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

CheckBox function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its Properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

ClassDefinition object

A ClassDefinition object is a PowerBuilder object that provides information about the class definition of a PowerBuilder object. You can examine a class in a PowerBuilder library or the class associated with an instantiated object.

All the properties are read-only. You cannot change the class definition using the ClassDefinition object. The ClassDefinition object has no events.

The ClassDefinition object lets you check:

The name of the class
The library the class was loaded from
The class definition of its ancestor, if any
The class definition of its parent or container object, if any
Whether the class is autoinstantiated
Whether the class is a system class (defined by PowerBuilder) or a user-defined object (defined in a PowerBuilder PBL)
The classes the object contains, such as the controls contained in a window
The variables and scripts defined in the class

Class names are always reported as lowercase, as you see them in the Browser.

Global functions and variables
Call FindFunctionDefinition to get a ScriptDefinition object describing the global function. Global variables are included in the VariableList array in the ClassDefinition object for the Application object.

Properties

ClassDefinition property	Datatype	Description
Ancestor	ClassDefinition	An object that represents the ancestor class. Ancestor is NULL when the ClassDefinition is describing PowerObject.
Category	TypeCategory	Specifies whether the type is simple, enumerated, or a class or structure. For a class definition, the value is ClassOrStructureType!.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

ClassDefinition property	Datatype	Description
DataTypeOf	String	The system class name of the object. DataTypeOf is a string representation of a value of the Object enumerated datatype. Values are lowercase with no exclamation point. Sample values include: window string any dropdownlistbox For objects you have defined, the datatype is the system class from which your object is inherited.
IsAutoinstantiate	Boolean	Indicates whether the class is an autoinstantiated class.
IsStructure	Boolean	Indicates whether the class is a structure.
IsSystemType	Boolean	Indicates whether the class is a system class—that is, one of the classes defined within PowerBuilder as opposed to a class defined in a PBL by a user.
IsVariableLength	Boolean	Specifies whether the datatype has a fixed size. Values are: TRUE – The datatype is variable length, meaning the datatype is a string, any, blob, or unbounded array. FALSE – The datatype is a fixed length.
IsVisualType	Boolean	Indicates whether the class is a visual (displayable) or non-visual type. Values are: TRUE – The class is visual, for example, a window or a control. FALSE – The class is non-visual, for example, a class user object or a simple datatype.
LibraryName	String	The fully qualified name of the library the class was loaded from.
Name	String	The name of the class. For a nested class, the name is returned in the form of *libraryEntryName`className*.
NestedClassList[ ]	ClassDefinition	An unbounded array of objects representing the nested classes and local structures for the object. The array is empty if there are no nested classes. Call the UpperBound function to find out the number of nested classes.
ParentClass	ClassDefinition	An object that represents the parent class that this class is nested within. The value is NULL if the class is not a nested class.

ClassDefinition property	Datatype	Description
ScriptList[ ]	ScriptDefinition	An unbounded array of objects representing the scripts implemented or defined in the collapsed class hierarchy. The array is empty if there are no scripts. Call the UpperBound function to find out the number of scripts.
VariableList[ ]	Variable Definition	An unbounded array of objects representing the properties or shared variables in the collapsed class hierarchy. The array is empty if there are no variables. Call the UpperBound function to find out the number of variables.

Functions

ClassDefinition function	Datatype returned	Description
ClassName	String	Returns the class of the object.
FindMatchingFunction	ScriptDefinition	Finds a function that matches the specified name and argument list.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

CommandButton control

You use a CommandButton to carry out an action. For example, you can use an OK button to confirm a deletion or a Cancel button to cancel the requested deletion.

Properties

Command Button property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.

Command Button property	Datatype	Description
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Move to the top. FALSE – Do not move to the top.
Cancel	Boolean	Specifies whether the control acts as the Cancel button. (The Cancel button receives a Clicked event if the user presses Esc.) Values are: TRUE – Acts as the Cancel button. FALSE – Does not act as the Cancel button.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Default	Boolean	Specifies whether the control is the default control. The default control has a thick border and receives a Clicked event if the user presses Enter without selecting a control. Values are: TRUE – Acts as the default. FALSE – Does not act as the default. **Editable controls** Default behavior can be affected by editable controls on the window. For more information, see the *PowerBuilder Users Guide*.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag Mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag Mode. FALSE – When the control is clicked, the control is not automatically in Drag Mode. You have to put the control into Drag Mode manually by using the Drag function.
DragIcon	String	Contains the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.

Command Button property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be clicked). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FlatStyle	Boolean	Specifies that the edge of the button displays only when the mouse hovers over it. This is the button style used in the Microsoft Rebar (coolbar) control. Values are: TRUE – Button has a flat appearance. FALSE – Button does not have a flat appearance.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Pointer	String	Specifies the name of the stock pointer of the file containing the pointer that is used for the control.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.

Command Button property	Datatype	Description
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

CommandButton event	Occurs
Clicked	When the control is clicked
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

CommandButton function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the control type of the control

Connection object (obsolete)

Obsolete object
Connection object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The Connection object specifies the parameters that PowerBuilder uses to connect to EAServer. You can customize the Connection object by defining a class user object inherited from the built-in Connection object. The user object has three events: Constructor, Destructor, and Error.

For more information about creating a custom Connection object, see the chapter on user objects in the *PowerBuilder Users Guide*.

For information about connecting to J2EE servers using the EJBConnection object, see *Application Techniques* and the *PowerBuilder Extension Reference*.

Properties

Connection property	Datatype	Description
Application	String	(Optional) Specifies the default package to be used for EAServer components. If you specify the default package in the Application property, you do not need to specify a package in the second parameter of the CreateInstance function.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ConnectString (obsolete)	String	Obsolete property. Used for distributed PowerBuilder connections.
Driver	String	The communications driver used for the connection.
ErrCode	Long	Code indicating the success or failure of the most recent operation. Values are: 0 – Success 50 – Distributed service error 52 – Distributed communications error 53 – Requested server not active 54 – Server not accepting requests 55 – Request terminated abnormally 56 – Response to request incomplete 57 – Not connected 58 – Object instance does not exist 62 – Server busy 75 – Server forced client to disconnect 80 – Server timed out client connection 87 – Connection to server has been lost 92 – Required property is missing or invalid
ErrText	String	Text indicating the success or failure of the most recent operation.
Handle	Objhandle	Internal use only.

Connection property	Datatype	Description
Location	String	Specifies the host name and port number for the EAServer server. Alternatively, the Location property can specify a fully-qualified URL that uses one of the following formats: iiop://*host*:*port* iiops://*host*:*port* http://*host*:*port* https://*host*:*port* To take advantage of EAServer's load balancing and failover support, you can also specify a semicolon-separated list of server locations (for example, "`iiop://srv1:9000;iiop://srv2:9000`").
Options	String	Specifies one or more communications options. If you specify more than one option, you need to separate the options with commas. EAServer clients can use the Options property of the Connection object to set ORB and SSL property values. Each ORB property value you specify must begin with `ORB`. For example, you can specify the path and name of the log file by specifying a value for ORBLogFile. See "Options" next.
Password	String	The password that will be used to connect to the server.
Trace (obsolete)	String	Obsolete property. Used for distributed PowerBuilder connections.
UserID	String	The name or ID of the user who will connect to the server.

Options

Option	Description
ORBHttp	Specifies whether the ORB should use HTTP tunneling to connect to the server. A setting of true specifies HTTP tunneling. The default is false. This parameter can also be set in an environment variable, JAG_HTTP. Some firewalls may not allow IIOP packets through, but almost all allow HTTP packets through. When connecting through such firewalls, set this property to false.

Option	Description
ORBIdleConnectionTimeout	Specifies the time, in seconds, that a connection is allowed to sit idle. When the timeout expires, the ORB closes the connection. The default is `0`, which specifies that connections can never time out. The connection timeout does not affect the life of proxy instance references; the ORB may close and reopen connections transparently between proxy method calls. Specifying a finite timeout for your client applications can improve server performance. If many instances of the client run simultaneously, a finite client connection timeout limits the number of server connections that are devoted to idle clients. A finite timeout also allows rebalancing of server load in an application that uses a cluster of servers.
ORBLogIIOP	Specifies whether the ORB should log IIOP protocol trace information. A setting of true enables logging. The default is false. This parameter can also be set in an environment variable, JAG_LOGIIOP. When this parameter is enabled, you must set the ORBLogFile option (or the corresponding environment variable) to specify the file where protocol log information is written.
ORBLogFile	Sets the path and name of the file to which to log client execution status and error messages. This parameter can also be set in an environment variable, JAG_LOGFILE. The default setting is `no log`.
ORBCodeSet	Sets the code set that the client uses. This parameter can also be set in an environment variable, JAG_CODESET. The default setting is `utf8`.
ORBRetryCount	Specifies the number of times to retry when the initial attempt to connect to the server fails. This parameter can also be set in an environment variable, JAG_RETRYCOUNT. The default is `5`.
ORBRetryDelay	Specifies the delay, in milliseconds, between retry attempts when the initial attempt to connect to the server fails. This parameter can also be set in an environment variable, JAG_RETRYDELAY. The default is `2000`.
ORBProxyHost	Specifies the machine name or the IP address of an SSL proxy.
ORBProxyPort	Specifies the port number of the SSL proxy.
ORBWebProxyHost	Specifies the host name or IP address of an HTTP proxy server that supports generic Web tunneling, sometimes called connect-based tunneling. There is no default for this property, and you must specify both the host name and port number properties. You can also specify the property by setting the environment variable JAG_WEBPROXYHOST.
ORBWebProxyPort	When generic Web tunneling is enabled by setting ORBWebProxyHost, specifies the port number at which the HTTP proxy server accepts connections. There is no default for this property, and you must specify both a host name and port. You can also specify the property by setting the environment variable JAG_WEBPROXYPORT.

Option	Description
ORBHttpExtraHeader	An optional setting to specify what extra information is appended to the header of each HTTP packet sent to a proxy server (specified with the ORBWebProxyHost parameter). You can also specify the property by setting the property JAG_HTTPEXTRAHEADER. There is no need to set this property unless you have configured the ORB to connect through an HTTP proxy server, and your HTTP proxy server has special protocol requirements. By default, the following line is appended to each packet: `User-agent: Jaguar/major.minor` where *major* and *minor* are the major and minor version numbers of your EAServer client software, respectively. You can set this property to specify text to be included at the end of each HTTP header. If multiple lines are included in the setting, they must be separated by carriage return and line feed characters. If the setting does not include a "`User-agent: `" line, then the default setting above is included in the HTTP header.
ORBsocketReuseLimit	Specifies the number of times a network connection can be reused to call methods from one server. The default is 0, which indicates no limit. The default is ideal for short-lived clients. The default may not be appropriate for a long-running client program that calls many methods from servers in a cluster. If sockets are reused indefinitely, the client may build an affinity for servers that it has already connected to rather than randomly distributing its server-side processing load among all the servers in the cluster. In these cases, the property should be tuned to best balance client performance against cluster load distribution. In Appeon testing, a setting of 10 to 30 proved to be a good starting point. If the reuse limit is too low, client performance degrades.
ORBcertificateLabel	Specifies the client certificate to use if the connection requires mutual authentication. The label is a simple name that identifies an X.509 certificate/private key in a PKCS #11 token. Required for mutual authentication.
ORBqop	Specifies the name of a security characteristic to use. Required for SSL.
ORBcacheSize	Specifies the size of the SSL session ID cache. Default is 100.
ORBpin	Specifies the PKCS #11 token PIN. This is required for logging in to a PKCS #11 token for client authentication and for retrieving trust information. Required for SSL.
ORBuserdata	Optional string that can be used to provide user-specified context information.
ORBentrustIniFile	Specifies the path name for the Entrust INI file that provides information on how to access Entrust. Required when the ORBuseEntrustid property is set to true.

Option	Description
ORBentrustUserProfile	Specifies the full path to the file containing an Entrust user profile. Optional when the Entrust single-login feature is available, required otherwise.
ORBuseEntrustID	Specifies whether to use the Entrust ID or the Sybase PKCS #11 token for authentication. This is a Boolean property. If set to FALSE, Sybase PKCS #11 token properties are valid and Entrust-specific properties are ignored. If set to true, Entrust-specific properties are valid and Sybase PKCS #11 token properties are ignored.
ORBentrustPassword	Specifies the password for logging in to Entrust with the specified user profile. Optional when the Entrust single-login feature is available, required otherwise.

Events

Connection event	Occurs
Constructor	When the Connection object is created
Destructor	When the Connection object is destroyed
Error	When a client request cannot be satisfied

Functions

Connection function	Datatype returned	Description
ClassName	String	Returns the class of the object
ConnectToServer (obsolete)	Long	Connects a client application to a server application
CreateInstance	Long	Creates an instance of a remote object on a server
DisconnectServer	Long	Disconnects a client application from a server application
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
GetServerInfo (obsolete)	Long	Obsolete function
Lookup (obsolete)	Long	Allows a PowerBuilder client to create an instance of an EAServer component **Obsolete function** Lookup is an obsolete function, because EAServer is no longer supported since PowerBuilder 2017.
PostEvent	Boolean	Adds an event to the end of the message queue for the object

Connection function	Datatype returned	Description
RemoteStopConnection (obsolete)	Long	Obsolete function
RemoteStopListening (obsolete)	Long	Obsolete function
TriggerEvent	Integer	Triggers a specified event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

ContextInformation object

The ContextInformation object provides information about an application's execution context, including current version information. Using this information, you can modify display characteristics and application behavior.

Properties

ContextInformation property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Handle	Long	Internal use only

Events

ContextInformation event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window

Functions

ContextInformation function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetCompanyName	Integer	Returns the company name for the current execution context.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetFixesVersion	Integer	Returns the fix level for the current PowerBuilder execution context.
GetHostObject	Integer	Provides a reference to the context's host object.
GetMajorVersion	Integer	Returns the major version for the current PowerBuilder execution context.
GetMinorVersion	Integer	Returns the minor version for the current PowerBuilder execution context.
GetName	Integer	Returns the name for the current execution context.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetShortName	Integer	Returns the short name for the current PowerBuilder execution context.
GetVersionName	Integer	Returns complete version information for the current PowerBuilder execution context.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

ContextKeyword object

The ContextKeyword object provides environment information for the current context. In the default environment, the ContextKeyword object provides host workstation environment variables.

Properties

ContextKeyword property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Handle	Long	Internal use only

Events

ContextKeyword event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window

Functions

ContextKeyword function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextKeywords	Integer	Retrieves one or more values associated with a specified keyword
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

CORBACurrent object (obsolete)

Obsolete object
CORBACurrent object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The CORBACurrent service object provides information about the EAServer transaction associated with a calling thread and enables the caller to control the transaction. The CORBACurrent object supports most of the methods defined by the CORBACurrent interface.

Properties

CORBACurrent property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

CORBACurrent event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

CORBACurrent function	Datatype returned	Description
BeginTransaction (obsolete)	Boolean	Creates a new transaction and associates it with the calling thread.
ClassName	String	Returns the class of the object.
CommitTransaction (obsolete)	Integer	Commits the transaction associated with the calling thread.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.

CORBACurrent function	Datatype returned	Description
GetStatus (obsolete)	Integer	Returns the status of the transaction associated with the calling thread.
GetTransactionName (obsolete)	String	Returns a string describing the transaction associated with the calling thread.
Init (obsolete)	Integer	Initializes an instance of the CORBACurrent service object.
PostEvent	Integer	Adds an event to the end of the message queue for the object.
ResumeTransaction (obsolete)	Integer	Associates the transaction passed in with the calling thread.
RollbackOnly (obsolete)	Integer	Modifies the transaction associated with the calling thread so that the outcome is to roll back the transaction.
RollbackTransaction (obsolete)	Integer	Rolls back the transaction associated with the calling thread.
SetTimeout (obsolete)	Boolean	Sets the timeout value for the top-level transaction. The transaction is rolled back if it does not complete before the timeout expires.
SuspendTransaction (obsolete)	Unsignedlong	Suspends the transaction associated with the calling thread and returns a handle to the transaction.
TriggerEvent	Integer	Triggers a specified event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

CORBAObject object (obsolete)

Obsolete object
CORBAObject object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The CORBAObject object gives PowerBuilder clients access to several standard CORBA methods. All proxy objects generated for EAServer components using the EAServer proxy generator are descendants of CORBAObject.

Properties

CORBAObject property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Events

CORBAObject event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

CORBAObject function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
_Is_A (obsolete)	Boolean	Checks to see whether a CORBA object is an instance of a class that implements a particular interface
_Narrow (obsolete)	Long	Converts a CORBA object reference from a general super-type to a more specific sub-type
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

DataStore object

A DataStore is a nonvisual DataWindow control. DataStores act just like DataWindow controls except that many of the visual properties associated with DataWindow controls do not apply to DataStores. Because you can print DataStores, PowerBuilder provides some events and functions for DataStores that pertain to the visual presentation of the data.

However, graph functions such as CategoryCount, CategoryName, GetData, SeriesCount, and so forth depend on the visual graph control, which is not created for a DataStore object. These functions return an error value or an empty string when used with DataStores.

Properties

DataStore property	Datatype	Description
DataObject	String	Specifies the name of the DataWindow or Report object associated with the control.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Object	DWObject	Used for the direct manipulation of objects within a DataWindow object from a script. These objects can be, for example, columns or text objects.

Events

Some but not all DataStore events have return codes that you can use to determine what action will be taken after the event occurs. You set the return codes in a RETURN statement in the event script.

DataStore event	Occurs
Constructor	Immediately before the Open event occurs in the window.
DBError	When a database error occurs in the DataStore. Return codes: 0 - (Default) Display the error message. 1 - Do not display the error message.
Destructor	Immediately after the Close event occurs in the window.
Error	When an error is found in a data or property expression for a DataWindow object.

DataStore event	Occurs
ItemChanged	When the AcceptText and Update functions are called for the DataStore. Return codes: 0 - (Default) Accept the data value. 1 - Reject the data value and do not allow focus to change. 2 - Reject the data value but allow focus to change.
ItemError	When a value imported into a DataStore from a string or file does not pass the validation rules for its column. Can also occur when the AcceptText and Update functions are called for the DataStore. Return codes: 0 - (Default) Reject the data value and show an error message box. 1 - Reject the data value with no message box. 2 - Accept the data value. 3 - Reject the data value but allow focus to change. If the return code is 0 or 1 (rejects the data), the field with the incorrect data regains the focus.
Printend	When the printing of the DataStore ends.
PrintPage	Before each page of the DataStore is formatted for printing. Return codes: 0 - Do not skip a page. 1 - Skip a page.
PrintStart	When the printing of the DataStore starts.
RetrieveEnd	When the retrieval for the DataStore is complete.
RetrieveRow	After a row has been retrieved. Return codes: 0 - (Default) Continue. 1 - Stop the retrieval.
RetrieveStart	When the retrieval for the DataStore is about to begin. Return codes: 0 - (Default) Continue. 1 - Do not perform the retrieval. 2 - Do not reset the rows and buffers before retrieving the data from the database.
SQLPreview	After a Retrieve, Update, or ReselectRow function call and immediately before the SQL statement is submitted to the DBMS. The following return codes specify the action that takes place when the event occurs after an Update function call only: 0 - (Default) Continue. 1 - Stop. 2 - Skip this request and execute the next request.
UpdateEnd	When all the updates from the DataStore to the database are complete.

DataStore event	Occurs
UpdateStart	After an Update function call and just before changes in the DataStore are sent to the database. Return codes: 0 - (Default) Continue. 1 - Do not perform the update.

Functions

DataStore function	Datatype returned	Description
AcceptText	Integer	Applies the contents of the DataStore's edit control to the current item in the DataStore buffer.
CategoryCount	Integer	Returns the number of categories in the specified graph. (Returns an error value or an empty string for DataStores.)
CategoryName	String	Returns the name of the specified category in the specified graph. (Returns an error value or an empty string for DataStores.)
ClassName	String	Returns the name assigned to the DataStore.
ClearValues	Integer	Deletes all items from the value list associated with the specified column in the DataStore.
Clipboard	Integer	Copies the specified graph in the DataStore to the clipboard.
CopyRTF	String	Returns the selected text, pictures, and input fields in a DataStore as a string with rich text formatting. Bitmaps and input fields are included in the string.
Create	Integer	Creates a DataWindow object using the specified source code and replaces the DataWindow object in the specified DataStore with the new DataWindow object.
CreateFrom	Integer	Creates a DataStore object from the passed ResultSet object.
DataCount	Long	Returns the number of data points in the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
DBCancel	Integer	Cancels a database retrieval in progress.
DeletedCount	Long	Returns the number of rows that have been deleted from the DataStore but have not yet been updated in the associated database table.
DeleteRow	Integer	Deletes the specified row from the DataStore.
Describe	String	Returns requested information about the structure of the DataStore.

DataStore function	Datatype returned	Description
Filter	Integer	Moves rows that do not meet the current filter criteria to the filter buffer.
FilteredCount	Integer	Returns the number of rows that do not meet the current filter criteria.
Find	Long	Returns the number of the first row that meets the search criteria within a specified search range in the detail area of a DataStore.
FindCategory	Integer	Returns the number of the specified category in the specified graph. (Returns an error value or an empty string for DataStores.)
FindGroupChange	Long	Searches starting at a specified row for the first break for the specified group in the DataStore.
FindRequired	Integer	Identifies the required columns that the user has not filled.
FindSeries	Integer	Returns the number of the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
GenerateHTMLForm	Integer	Creates an HTML Form element containing columns for one or more rows. Also returns an HTML Style element containing style sheet information.
GenerateResultSet	Long	Returns an EAServer result set from a PowerBuilder user object running as a component on EAServer. **Obsolete function** GenerateResultSet is an obsolete function, because EAServer is no longer supported since PowerBuilder 2017.
GetBorderStyle	Border (enumerated)	Returns a Border enumerated datatype indicating the border style of the specified column in the DataStore. Border enumerated datatypes are: Box! Lowered! NoBorder! Raised! ResizeBorder! ShadowBox! Underline!
GetChanges	Long	Retrieves changes made to a DataStore into a blob. This function is used primarily in distributed applications.
GetChild	Integer	Stores in the specified variable the name of the child DataWindow in the specified column.
GetClickedColumn	Integer	Obtains the number of the column the user clicked or double-clicked in a DataStore.
GetClickedRow	Long	Obtains the number of the row the user clicked or double-clicked in a DataStore.
GetColumn	Integer	Returns the number of the current column in the DataStore.

DataStore function	Datatype returned	Description
GetColumnName	String	Returns the name of the current column in the DataStore.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetData	Double	Returns the value of the specified data in the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
GetDataPieExplode	Integer	Reports the percentage that a pie slice is exploded in a pie graph. (Returns an error value or an empty string for DataStores.)
GetDataStyle	Integer	Finds out the appearance of a data point in a graph. (Returns an error value or an empty string for DataStores.)
GetDataValue	Integer	Returns the value of the specified data in the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
GetFormat	String	Returns the format used for display in the specified column of the DataStore.
GetFullState	Long	Retrieves the complete state of a DataStore into a blob. This function is used primarily in distributed applications.
GetItemDate	Date	Returns the date data in the specified row and column of the DataStore.
GetItemDateTime	DateTime	Returns the datetime data in the specified row and column of the DataStore.
GetItemDecimal	Decimal	Returns the decimal data in the specified row and column of the DataStore.
GetItemNumber	Double	Returns the numeric data in the specified row and column of the DataStore.
GetItemStatus	dwItemStatus (enumerated)	Returns the status of the item at the specified row and column location in the specified buffer. Values are: DataModified! New! NewModified! NotModified!
GetItemString	String	Returns the string data in the specified row and column of the DataStore.
GetItemTime	Time	Returns the time data in the specified row and column of the DataStore.
GetNextModified	Long	Returns the number of the first row that was modified in the specified buffer in the specified DataStore after the specified row.
GetParent	PowerObject	Returns a reference to the name of the parent object.

DataStore function	Datatype returned	Description
GetRow	Long	Returns an integer containing the number of the current row in the DataStore.
GetRowFromRowId	Long	Gets the row number of a row in a DataStore from the unique row identifier associated with that row.
GetRowIdFromRow	Long	Gets the unique row identifier of a row in a DataStore from the row number associated with that row.
GetSelectedRow	Integer	Returns the number of the first selected row after the specified row number in the DataStore.
GetSeriesStyle	Integer	Finds out the appearance of a series in a graph. (Returns an error value or an empty string for DataStores.)
GetSQLSelect	String	Returns the current SELECT statement for the DataStore.
GetStateStatus	Long	Retrieves the current status of the internal state flags for a DataWindow and places this information in a blob. This function is used primarily in distributed applications.
GetText	String	Returns the text in the edit control over the current row and column of the DataStore.
GetTrans	Integer	Returns the values in the DataStore Transaction object.
GetValidate	String	Returns the validation rule used in the specified column of the DataStore.
GetValue	String	Returns the specified item in the value list for the specified column.
GroupCalc	Integer	Recalculates the breaks in the groups in the DataStore.
ImportClipboard	Long	Copies data from the clipboard to the DataStore.
ImportFile	Long	Copies data from a file to the DataStore.
ImportString	Long	Copies data from a string to the DataStore.
InsertDocument	Integer	Inserts a rich text format or plain text file into a DataStore. You use a function parameter to specify how the new content is added: • It can be inserted at the insertion point • It can replace all existing content This function applies only to DataStores whose content has the RichText presentation style.
InsertRow	Long	Inserts a new initialized row before the specified row in the DataStore.
IsSelected	Boolean	Returns true if the specified row in the DataStore is selected; returns false if the row is not selected or is greater than the number of rows in the DataStore.

DataStore function	Datatype returned	Description
ModifiedCount	Long	Returns the number of rows that have been modified in the DataStore but have not yet been updated in the associated database table.
Modify	String	Uses the specification contained in a string to modify the DataStore.
PasteRTF	Long	Pastes rich text data from a string into a DataStore whose content has the RichText presentation style.
PostEvent	Boolean	Adds an event to the end of the message queue for the DataStore.
Print	Integer	Sends the content of the DataStore to the current printer. This function has several syntaxes.
PrintCancel	Integer	Cancels the print job and deletes the spool file (if any) when the content of the DataStore is sent to print. This function has two syntaxes. Use Syntax 1 when Syntax 1 of the Print function was used to send it to print.
ReselectRow	Integer	Accesses the database to reselect all columns that can be updated and refreshes all timestamp columns in a row in the DataStore.
Reset	Integer	Clears all the data from a DataStore.
ResetDataColors	Integer	Resets the color of a data point to the color specified for the series. (Returns an error value or an empty string for DataStores.)
ResetTransObject	Integer	Stops the DataStore from using a programmer-defined Transaction object (thereafter, the DataStore uses its internal Transaction object).
ResetUpdate	Integer	Resets the update flags for the DataStore.
Retrieve	Long	Causes the DataStore to retrieve rows from the database.
RowCount	Long	Returns the number of rows currently available in the DataStore (all the rows retrieved minus any deleted rows plus any inserted rows minus any rows that have been filtered out).
RowsCopy	Integer	Copies a range of rows from one DataStore to another DataStore (or DataWindow control) or from one buffer to another within a single DataStore.
RowsDiscard	Integer	Discards a range of rows. The rows cannot be restored unless retrieved from the database.
RowsMove	Integer	Clears a range of rows from a DataStore and inserts the rows in another DataStore (or DataWindow control) or another buffer of the same DataStore.
SaveAs	Integer	Saves the content of the DataStore to the specified file, in the specified format, with or without column headings at the beginning
SaveAsAscii	Long	Saves the content of a DataStore into a standard ASCII text file.

DataStore function	Datatype returned	Description
SelectRow	Integer	Selects or deselects the specified row of the DataStore.
SeriesCount	Integer	Returns the number of series in the specified graph. (Returns an error value or an empty string for DataStores.)
SeriesName	String	Returns the name of the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
SetBorderStyle	Integer	Sets the border style of the specified column in the DataStore.
SetChanges	Long	Applies changes captured with GetChanges to a DataStore. This function is used primarily in distributed applications.
SetColumn	Integer	Makes the specified column the current column in the DataStore.
SetDataPieExplode	Integer	Explodes a pie slice in a pie graph. (Returns an error value or an empty string for DataStores.)
SetDataStyle	Integer	For the specified data point in the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
SetDetailHeight	Integer	Sets the height of each row in a specified range.
SetFilter	Integer	Defines the filter criteria for the DataStore. The actual filtering is performed by the Filter function.
SetFormat	Integer	Sets the display format for the specified column of the DataStore.
SetFullState	Long	Applies the contents of a DataWindow blob retrieved by GetFullState to a DataStore. This function is used primarily in distributed applications.
SetHTMLAction	Integer	Accepts action and context information about user interaction with the Web DataWindow client control in a Web browser so that newly generated HTML can reflect any requested changes. **Obsolete function** SetHTMLAction is an obsolete function, because Web DataWindow technology is obsolete.
SetItem	Integer	Sets the value of the specified row and column of the specified DataStore.
SetItemStatus	Integer	Sets the status of a row in a specified column of the DataStore in the specified buffer.
SetPosition	Integer	Moves an object within the DataStore to another band or changes the front-to-back order of objects within a band.
SetRow	Integer	Makes the specified row the current row in the DataStore.
SetSeriesStyle	Integer	For the specified series in the specified graph. (Returns an error value or an empty string for DataStores.)
SetSort	Integer	Defines the sort criteria for the DataStore. The actual sorting is performed by the Sort function.
SetSQLPreview	Integer	Sets the current SQL statement for the DataStore.
SetSQLSelect	Integer	Changes the current SELECT statement for the DataStore.

DataStore function	Datatype returned	Description
SetText	Integer	Replaces the text in the edit control at the current row and column of the DataStore with the specified text.
SetTrans	Integer	Sets values in the DataStore's internal Transaction object.
SetTransObject	Integer	Sets the Transaction object for the DataStore and provides control over the transaction, including the ability to commit from a script.
SetValidate	Integer	Changes the validation rule used for the specified column of the DataStore.
SetValue	Integer	Sets the value of the specified item in the value list or the code table of the specified column of the DataStore.
SetWSObject	Integer	Causes a DataStore (or DataWindow control) to use a programmer-specified connection object. The connection object provides the information necessary for communicating with a Web service data source.
ShareData	Integer	Shares data between a primary DataStore (or DataWindow control) and a secondary DataStore (or DataWindow control).
ShareDataOff	Integer	Turns off sharing for the DataStore. If the DataStore is primary, all secondary DataStores (or DataWindow controls) are disconnected and their DataWindow objects no longer contain data.
Sort	Integer	Sorts the rows of the DataStore based on its current sort criteria.
TriggerEvent	Integer	Triggers a specified event in the DataStore and executes the script for the event.
TypeOf	Object	Returns the type of the DataStore.
Update	Integer	Sends to the database all inserts, deletes, and updates of the DataStore.

DataWindow control

You place DataWindow *controls* in a window or user object and then specify the DataWindow *object* you want to use within them to display and manipulate data in the window.

A DataWindow object allows users to display, manipulate, and update database or other information. You build DataWindow objects in the DataWindow painter.

For information about DataWindow objects, see the *PowerBuilder Users Guide*.

Obsolete functions
Several DataWindow control functions are described as obsolete, which means that although the function operates as usual in this release, it will be removed in a future release. You should replace all use of these functions as soon as possible.

Properties

DataWindow property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the border style of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ControlMenu	Boolean	Specifies whether the Control Menu box displays in the control title bar. Values are: TRUE – Control Menu box displays in the control title bar. FALSE – Control Menu box does not display in the control title bar.
DataObject	String	Specifies the name of the DataWindow object associated with the control.

DataWindow property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag Mode. DragAuto has these boolean values: TRUE – When the control is clicked, the control is automatically in Drag Mode. FALSE – When the control is clicked, the control is not automatically in Drag Mode. You have to put the control into Drag Mode manually by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
Height	Integer	Specifies the height of the DataWindow control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays in the control when all the data cannot be displayed at one time. Values are: TRUE – Horizontal scroll bar is displayed. FALSE – Horizontal scroll bar is not displayed.
HSplitScroll	Boolean	Specifies whether the split bar displays in the control. Values are: TRUE – Split bar is displayed. FALSE – Split bar is not displayed.
Icon	String	Specifies the name of the ICO file that contains the icon that displays when the DataWindow control is minimized.
LiveScroll	Boolean	Scrolls the rows in the DataWindow control while the user is moving the scroll box.
MaxBox	Boolean	Specifies whether a Maximize Box displays in the DataWindow control title bar. Values are: TRUE – Maximize Box displays. FALSE – Maximize Box does not display.
MinBox	Boolean	Specifies whether a Minimize Box displays in the DataWindow control title bar. Values are: TRUE – Minimize Box displays. FALSE – Minimize Box does not display.

DataWindow property	Datatype	Description
Object	DWObject	Used for the direct manipulation of objects within a DataWindow object from a script. These objects can be, for example, columns or text objects.
Resizable	Boolean	Specifies whether the DataWindow control is resizable. Values are: TRUE – DataWindow is resizable. FALSE – DataWindow is not resizable.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the DataWindow control within the window or user object (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the DataWindow control.
Title	String	Specifies the text that displays in the DataWindow control title bar.
TitleBar	Boolean	Specifies whether a title bar displays in the DataWindow control. The user can move the DataWindow control only if it has a title bar. Values are: TRUE – Title bar is displayed in control. FALSE – No title bar is displayed in control.
Visible	Boolean	Specifies whether the DataWindow control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar displays in the control when not all the data can be displayed at one time. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Width	Integer	Specifies the width of the DataWindow control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top edge of the window), in PowerBuilder units.

Events

Some but not all DataWindow events have return codes that you can use to determine what action is taken after the event occurs. You set the return codes in a RETURN statement in the event script.

DataWindow event	Occurs
ButtonClicked	When the user clicks a button.
ButtonClicking	When the user clicks a button. This event occurs before the ButtonClicked event.
Clicked	When the user clicks between fields in the DataWindow control. Return codes: 0 - (Default) Continue processing. 1 - Stop processing.
Collapsed	When a node in a TreeView DataWindow has collapsed.
Collapsing	Before a node in a TreeView DataWindow collapses.
Constructor	Immediately before the Open event occurs in the window.
DBError	When a database error occurs in the DataWindow control.
Destructor	Immediately after the Close event occurs in the window.
DoubleClicked	When the user double-clicks between fields in the DataWindow control. For a RichText presentation style DataWindow, when the user double-clicks in the text.
DragDrop	When a dragged control is dropped on the DataWindow control.
DragEnter	When a dragged control enters the DataWindow control.
DragLeave	When a dragged control leaves the DataWindow control.
DragWithin	When a dragged control is within the DataWindow control.
EditChanged	When a user types in an edit control in the DataWindow control.
Error	When an error is found in a data or property expression for a DataWindow object.
Expanded	When a node in a TreeView DataWindow has expanded.
Expanding	Before a node in a TreeView DataWindow expands.
GetFocus	Just before the DataWindow control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
ItemChanged	When a field in the DataWindow has been modified and loses focus (for example, the user presses Enter, the Tab key, or an arrow key, or clicks the mouse on another field *within the DataWindow*). Return codes: 0 - (Default) Accept the data value. 1 - Reject the data value and do not allow focus to change. 2 - Reject the data value but allow focus to change.

DataWindow event	Occurs
ItemError	When a field has been modified, the field loses focus (for example, the user presses Enter, Tab, or an arrow key, or clicks the mouse on another field), and the field does not pass the validation rules for its column. Return codes: 0 - (Default) Reject the data value and show an error message box. 1 - Reject the data value with no message box. 2 - Accept the data value. 3 - Reject the data value but allow focus to change. If the Return code is 0 or 1 (rejects the data), the field with the incorrect data regains the focus.
ItemFocusChanged	When the current item in the control changes.
LoseFocus	When the DataWindow control loses focus (becomes inactive).
Other	When a Windows message occurs that is not a PowerBuilder event.
Printend	When the printing of the DataWindow ends.
PrintPage	Before each page of the DataWindow is formatted for printing. Return codes: 0 - Do not skip a page. 1 - Skip a page.
PrintStart	When the printing of the DataWindow starts.
RButtonDown	When the right mouse button is pressed on the control. For a RichText presentation style DataWindow, if PopUp Menu has been turned on, this event is not triggered when the right mouse button is pressed.
Resize	When the user or a script resizes a DataWindow control.
RetrieveEnd	When the retrieval for the DataWindow is complete.
RetrieveRow	After a row has been retrieved. Return codes: 0 - (Default) Continue. 1 - Stop the retrieval.
RetrieveStart	When the retrieval for the DataWindow is about to begin. Return codes: 0 - (Default) Continue. 1 - Do not perform the retrieval. 2 - Do not reset the rows and buffers before retrieving the data from the database.
RowFocusChanged	When the current row changes in the DataWindow.
RowFocusChanging	When the current row is about the change in the DataWindow. This event occurs before the RowFocusChanged event.
ScrollHorizontal	When the user scrolls right or left in the DataWindow control with the Tab or arrow keys or the scroll bar.

DataWindow event	Occurs
ScrollVertical	When the user scrolls up or down in the DataWindow control with the Tab or arrow keys or the scroll bar.
SQLPreview	After a Retrieve, Update, or ReselectRow function call and immediately before the SQL statement is submitted to the DBMS. The following return codes specify the action that takes place when the event occurs after an Update function call only: 0 - (Default) Continue. 1 - Stop. 2 - Skip this request and execute the next request.
UpdateEnd	When all the updates from the DataWindow to the database are complete.
UpdateStart	After an Update function call and just before changes in the DataWindow are sent to the database. Return codes: 0 - (Default) Continue. 1 - Do not perform the update.
WSError	Occurs when an error is returned for a DataWindow using the WSConnection object to connect to a Web service data source.

Functions

DataWindow function	Datatype returned	Description
AcceptText	Integer	Applies the contents of the DataWindow control's edit control to the current item in the DataWindow buffer.
CanUndo	Boolean	Specifies whether the last edit can be undone with the Undo function. Applies to the edit control over the current row and column. Values are: TRUE – Last edit can be undone. FALSE – Last edit cannot be undone.
CategoryCount	Integer	Returns the number of categories in the specified graph in the DataWindow control.
CategoryName	String	Returns the name of the specified category in the specified graph in the DataWindow control.
ClassName	String	Returns the name assigned to the DataWindow control.
Clear	Integer	Clears (deletes) the selected text in the edit control of the DataWindow control. For a RichText presentation style DataWindow, clears the selected text in the DataWindow.

DataWindow function	Datatype returned	Description
ClearValues	Integer	Deletes all items from the value list associated with the specified column in the DataWindow control.
Clipboard	Integer	Copies the specified graph in the DataWindow control to the clipboard.
Copy	Integer	Copies the selected text in the edit control over the current row and column of the DataWindow control to the clipboard. For a RichText presentation style DataWindow, copies the selected text in the DataWindow control.
CopyRTF	String	Returns the selected text, pictures, and input fields in a DataWindow control as a string with rich text formatting. Bitmaps and input fields are included in the string.
Create	Integer	Creates a DataWindow object using the specified source code and replaces the DataWindow object in the specified DataWindow control with the new DataWindow object.
CrosstabDialog	Integer	Displays the Crosstab Definition dialog box so the user can modify the definition of a crosstab DataWindow object during execution.
Cut	Integer	Cuts the selected text from the edit control over the current row and column of the DataWindow and stores it in the clipboard. For a RichText presentation style DataWindow, cuts the selected text in the DataWindow control.
DataCount	Long	Returns the number of data points in the specified series in the specified graph in the DataWindow control.
DBCancel	Integer	Cancels a database retrieval in progress.
DBErrorCode	Long	Returns the error code (number) generated by a database error. **Obsolete function** DBErrorCode is an obsolete function and will be discontinued in a future release. Database error codes are now available as event arguments.
DBErrorMessage	String	Returns a string containing the text of the error message generated by a database error. **Obsolete function** DBErrorMessage is an obsolete function and will be discontinued in a future release. Database error messages are now available as event arguments.
DeletedCount	Long	Returns the number of rows that have been deleted from the DataWindow control but have not yet been updated in the associated database table.
DeleteRow	Integer	Deletes the specified row from the DataWindow control.
Describe	String	Returns requested information about the structure of the DataWindow control.
Drag	Integer	Starts or ends the dragging of the DataWindow control.

DataWindow function	Datatype returned	Description
Filter	Integer	Displays specific rows of the DataWindow control based on its current filter.
FilteredCount	Integer	Returns the number of rows that are not visible because of the current filter.
Find	Long	Syntax 1: Finds the next row in a DataWindow control in which data meets a specified condition. Syntax 2: For Rich Text presentation style Data Windows, finds the specified text in the control and highlights the text if found. You can specify search direction and whether to match whole words and case.
FindCategory	Integer	Returns the number of the specified category in the specified graph in the DataWindow control.
FindGroupChange	Long	Searches starting at a specified row for the first break for the specified group in the DataWindow control.
FindNext	Integer	Finds the next occurrence of text in the control and highlights it, using criteria set up in a previous call of the Find function. This function applies only to DataWindow controls whose content has the RichText presentation style.
FindRequired	Integer	Identifies the required columns that the user has not filled.
FindSeries	Integer	Returns the number of the specified series in the specified graph in the DataWindow control.
GenerateHTMLForm	Integer	Creates an HTML Form element containing columns for one or more rows. Also returns an HTML Style element containing style sheet information.
GenerateResultSet	Long	Returns an EAServer result set from a PowerBuilder user object running as a component on EAServer. **Obsolete function** GenerateResultSet is an obsolete function, because EAServer is no longer supported since PowerBuilder 2017.
GetBandAtPointer	String	Returns the string containing the band in which the pointer is currently located followed by a tab character (~t) and the number of the row associated with the band.

DataWindow function	Datatype returned	Description
GetBorderStyle	Border (enumerated)	Returns a Border enumerated datatype indicating the border style of the specified column in the DataWindow control. Values are: Box! Lowered! NoBorder! Raised! ResizeBorder! ShadowBox! Underline!
GetChanges	Long	Retrieves changes made to a DataWindow into a blob. This function is used primarily in distributed applications.
GetChild	Integer	Stores in the specified variable the name of the child DataWindow in the specified column.
GetClickedColumn	Integer	Returns the number of the column in the DataWindow control that the user clicked or double-clicked.
GetClickedRow	Long	Returns the number of the row in the DataWindow control that the user clicked or double-clicked.
GetColumn	Integer	Returns the number of the current column in the DataWindow control.
GetColumnName	String	Returns the name of the current column in the DataWindow control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetData	Double	Returns the value of the specified data in the specified series in the specified graph in the DataWindow control. See also GetDataValue.
GetDataPieExplode	Integer	Reports the percentage that a pie slice is exploded in a pie graph.
GetDataLabelling	Integer	Determines whether the data at a given data point is labeled in a DirectX 3D graph.
GetDataStyle	Integer	Finds out the appearance of a data point in a graph. Each data point in a series can have individual appearance settings. There are different syntaxes, depending on what settings you want to check.
GetDataTransparency	Integer	Obtains the transparency percentage of a series in a DirectX 3D graph.
GetDataValue	Integer	Returns the value of the specified data in the specified series in the specified graph in the DataWindow control.
GetFormat	String	Returns the format used for display in the specified column of the DataWindow control.
GetFullState	Long	Retrieves the complete state of a DataWindow into a blob. This function is used primarily in distributed applications.

DataWindow function	Datatype returned	Description
GetItemDate	Date	Returns the date data in the specified row and column of the DataWindow control.
GetItemDateTime	DateTime	Returns the datetime data in the specified row and column of the DataWindow control.
GetItemDecimal	Decimal	Returns the decimal data in the specified row and column of the DataWindow control.
GetItemNumber	Double	Returns the numeric data in the specified row and column of the DataWindow control.
GetItemStatus	dwItemStatus (enumerated)	Returns the status of the item at the specified row and column location in the specified buffer. Values are: DataModified! New! NewModified! NotModified!
GetItemString	String	Returns the string data in the specified row and column of the DataWindow control.
GetItemTime	Time	Returns the time data in the specified row and column of the DataWindow control.
GetMessageText	String	Obtains the message text generated by a crosstab DataWindow object in the DataWindow control. **Obsolete function** GetMessageText is an obsolete function and will be discontinued in a future release. Message text is now available as an argument in a user-defined DataWindow event associated with the token pbm_dwnmessagetext.
GetNextModified	Long	Returns the number of the first row that was modified in the specified buffer in the specified DataWindow control after the specified row.
GetObjectAtPointer	String	Returns the string containing the name of the column or graphic control under the pointer in the DataWindow control, then a tab character (~t), and then the row number.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetRow	Long	Returns an integer containing the number of the current row in the DataWindow control.
GetRowFromRowId	Long	Gets the row number of a row in a DataWindow control from the unique row identifier associated with that row.
GetRowIdFromRow	Long	Gets the unique row identifier of a row in a DataWindow control from the row number associated with that row.
GetSelectedRow	Integer	Returns the number of the first selected row after the specified row number in the DataWindow control.
GetSeriesLabelling	Integer	Obtains the series labelling for a DirectX 3D graph.

DataWindow function	Datatype returned	Description
GetSeriesStyle	Integer	Finds out the appearance of a series in a graph. There are several syntaxes, depending on what settings you want.
GetSeriesTransparency	Integer	Obtains the transparency percentage of a series in a DirectX 3D graph.
GetSQLPreview	String	Returns the current SQL statement the DataWindow control is submitting to the database. **Obsolete function** GetSQLPreview is an obsolete function and will be discontinued in a future release. SQL syntax is now available as an event argument.
GetSQLSelect	String	Returns the current SELECT statement for the DataWindow control.
GetStateStatus	Long	Retrieves the current status of the internal state flags for a DataWindow and places this information in a blob. This function is used primarily in distributed applications.
GetText	String	Returns the text in the edit control over the current row and column of the DataWindow control.
GetTrans	Integer	Returns the values in the DataWindow Transaction object.
GetUpdateStatus	Integer	Stores the number of the row that will be updated in a variable, and the dwBuffer enumerated datatype identifying the buffer containing the row in another variable. **Obsolete function** GetUpdateStatus is an obsolete function and will be discontinued in a future release. Update status is now available as an argument in the DataWindow control DBError and SQLPreview events.
GetValidate	String	Returns the validation rule used in the specified column of the DataWindow control.
GetValue	String	Returns the specified item in the value list for the specified column.
GroupCalc	Integer	Recalculates the breaks in the groups in the DataWindow control.
Hide	Integer	Makes the control invisible.
ImportClipboard	Long	Copies data from the clipboard to the DataWindow control.
ImportFile	Long	Copies data from a file to the DataWindow control.
ImportString	Long	Copies data from a string to the DataWindow control.
InsertDocument	Integer	Inserts a rich text format or plain text file into a DataWindow control. You use a function parameter to specify how the new content is added: • It can be inserted at the insertion point. • It can replace all existing content. This function applies only to DataWindow controls whose content has the RichText presentation style.

DataWindow function	Datatype returned	Description
InsertRow	Long	Inserts a new initialized row before the specified row in the DataWindow control.
IsSelected	Boolean	Returns true if the specified row in the DataWindow is selected; returns false if the row is not selected or is greater than the number of rows in the DataWindow control.
LineCount	Integer	Determines the number of lines in an edit control that allows multiple lines.
ModifiedCount	Long	Returns the number of rows that have been modified in the DataWindow control but have not yet been updated in the associated database table.
Modify	String	Uses the specification contained in a string to modify the DataWindow control.
Move	Integer	Moves the specified DataWindow control to a specified location.
ObjectAtPointer	grObjectType	Returns the number of the series the pointer is over and the number of the data point in the graph in the DataWindow control and identifies the object type.
OLEActivate	Integer	Activates OLE for the OLE object in the specified row and column of the DataWindow control.
Paste	Integer	Inserts the contents of the clipboard into the edit control over the current row and column in the DataWindow control.
PasteRTF	Long	Pastes rich text data from a string into a DataWindow control whose content has the RichText presentation style.
PointerX	Integer	Returns the distance of the pointer from the left edge of the DataWindow control.
PointerY	Integer	Returns the distance of the pointer from the top of the DataWindow control.
Position	Integer	Syntax 1: Reports the insertion point's position in the edit control over the current row and column of the DataWindow control. Syntax 2: Reports the line and column position of the insertion point or the start and end of selected text in a DataWindow control whose content has the RichText presentation style.
PostEvent	Boolean	Adds an event to the end of the message queue for the DataWindow control.
Print	Integer	Sends the content of the DataWindow control to the current printer. This function has several syntaxes.
PrintCancel	Integer	Cancels the print job and deletes the spool file (if any) when the content of the DataWindow control is sent to print. This function has two syntaxes. Use Syntax 1 when Syntax 1 of the Print function is used to send the content of the DataWindow control to the current printer.

DataWindow function	Datatype returned	Description
ReplaceText	Integer	Replaces the selected text in the edit control with the specified string.
ReselectRow	Integer	Accesses the database to reselect all columns that can be updated and refreshes all timestamp columns in a row in the DataWindow control.
Reset	Integer	Clears all the data from a DataWindow control.
Reset	Integer	Deletes the data, the categories, or the series from a graph within a DataWindow object with an external data source.
ResetDataColors	Integer	Resets the color of a data point to the color specified for the series.
ResetTransObject	Integer	Stops the DataWindow control from using a programmer-defined Transaction object (thereafter, the DataWindow uses its internal Transaction object).
ResetUpdate	Integer	Resets the update flags for the DataWindow control.
Resize	Integer	Changes the width and height of the DataWindow control.
Retrieve	Long	Retrieves rows from the database for the DataWindow control.
RowCount	Long	Returns the number of rows currently available in the DataWindow control (all the rows retrieved minus any deleted rows plus any inserted rows minus any rows that have been filtered out).
RowsCopy	Integer	Copies a range of rows from one DataWindow control to another or from one buffer to another within a single DataWindow control.
RowsDiscard	Integer	Discards a range of rows. The rows cannot be restored unless retrieved from the database.
RowsMove	Integer	Clears a range of rows from a DataWindow control and inserts the rows in another DataWindow control or another buffer of the same DataWindow control.
SaveAs	Integer	Saves the data represented in the specified graph in the DataWindow control to the specified file, in the specified format.
SaveAs	Integer	Saves the content of the DataWindow control to the specified file, in the specified format, with or without column headings at the beginning.
SaveAsAscii	Long	Saves the content of a DataWindow into a standard ASCII text file.
Scroll	Integer	Scrolls the edit control of a DataWindow control in the specified direction the specified number of lines.
ScrollNextPage	Long	Syntax 1: Scrolls forward by the number of rows showing in the DataWindow (when the DataWindow control contents does not have the RichText presentation style). Syntax 2: Scrolls to the next page of the document in a DataWindow control whose content has the RichText presentation style.

DataWindow function	Datatype returned	Description
ScrollNextRow	Long	Scrolls the DataWindow control to the next row. ScrollNextRow changes the current row but not the current column.
ScrollPriorPage	Long	Syntax 1: Scrolls backward by the number of rows showing in the DataWindow (when the DataWindow control content does not have the RichText presentation style). Syntax 2: Scrolls to the prior page of the document in a DataWindow control whose content has the RichText presentation style.
ScrollPriorRow	Long	Scrolls to the previous row. The ScrollPriorRow function changes the current row in the DataWindow control but does not change the current column.
ScrollToRow	Integer	Causes the control to scroll to the specified row. ScrollToRow changes the current row in the DataWindow control but does not change the current column.
SelectedLength	Integer	Reports the total number of characters and spaces (length) in the selected text in the edit control over the current row and column.
SelectedLine	Integer	Reports the line number in the edit control over the current row and column.
SelectedStart	Integer	Reports the starting position in the edit control over the current row and column.
SelectedText	String	Reports what text (if any) is selected in the edit control over the current row and column of the DataWindow control.
SelectRow	Integer	Selects or deselects the specified row in the DataWindow control.
SelectText	Integer	Syntax 1: Selects text in the edit control of a DataWindow control (other than one whose content is in the RichText presentation style). You specify where the selection begins and how many characters to select. Syntax 2: Selects text beginning and ending at the specified line and character positions in a DataWindow control whose content is in the RichText presentation style.
SelectTextAll	Integer	Selects all the content of a DataWindow control with the RichText presentation style.
SelectTextLine	Integer	Selects the line containing the insertion point in a DataWindow control with the RichText presentation style.
SelectTextWord	Integer	Selects the word containing the insertion point in a DataWindow control with the RichText presentation style.
SeriesCount	Integer	Returns the number of series in the specified graph in the DataWindow control.
SeriesName	String	Returns the name of the specified series in the specified graph in the DataWindow control.

DataWindow function	Datatype returned	Description
SetActionCode	Integer	Defines the action a DataWindow control takes following an event. **Obsolete function** SetActionCode is an obsolete function and will be discontinued in a future release. You now set return codes in a return statement in the event script.
SetBorderStyle	Integer	Sets the border style of the specified column in the DataWindow control.
SetChanges	Long	Applies changes captured with GetChanges to a DataWindow. This function is used primarily in distributed applications.
SetColumn	Integer	Makes the specified column the current column in the DataWindow control.
SetDataPieExplode	Integer	Explodes a pie slice in a pie graph.
SetDataLabelling	Integer	Sets the series label for a DirectX 3D graph.
SetDataStyle	Integer	For the specified data point in the specified series in the specified graph in the DataWindow control: Syntax 1: Sets the data point's color. Syntax 2: Sets the line style and width for the data point. Syntax 3: Sets the fill pattern or symbol for the data point.
SetDataTransparency	Integer	Sets the transparency percentage for a data point in a series in a DirectX 3D graph.
SetDetailHeight	Integer	Sets the height of each row in a specified range.
SetFilter	Integer	Defines the filter criteria for the DataWindow control. The actual filtering is performed by the Filter function.
SetFocus	Integer	Sets focus to the DataWindow control.
SetFormat	Integer	Sets the display format for the specified column of the DataWindow control.
SetFullState	Long	Applies the contents of a DataWindow blob retrieved by GetFullState to a DataWindow. This function is used primarily in distributed applications.
SetHTMLAction	Integer	Accepts action and context information about user interaction with the Web DataWindow client control in a Web browser so that newly generated HTML can reflect any requested changes. **Obsolete function** SetHTMLAction is an obsolete function, because Web DataWindow technology is obsolete.
SetItem	Integer	Sets the value of the specified row and column of the specified DataWindow control.
SetItemStatus	Integer	Sets the status of a row in a specified column of the DataWindow control in the specified buffer.

DataWindow function	Datatype returned	Description
SetPosition	Integer	Syntax 1: Specifies whether the DataWindow control always displays on top in the front-to-back order within the window. Syntax 2: Moves an object within the DataWindow to another band or changes the front-to-back order of objects within a band.
SetRedraw	Integer	Controls automatic redrawing of the DataWindow control after each change in its properties or contents.
SetRow	Integer	Makes the specified row the current row in the DataWindow control.
SetRowFocusIndicator	Integer	Sets the current row indicator for the DataWindow control.
SetSeriesLabelling	Integer	Sets the series label for a DirectX 3D graph.
SetSeriesStyle	Integer	For the specified series in the specified graph in the DataWindow control: Syntax 1: Sets the series' color. Syntax 2: Sets the linestyle and width. Syntax 3: Sets the fill pattern or symbol for data markers in the series. Syntax 4: Specifies that the series is an overlay.
SetSort	Integer	Defines the sort criteria for the DataWindow control. The actual sorting is performed by the Sort function.
SetSeriesTransparency	Integer	Sets the transparency percentage of a series in a DirectX 3D type graph.
SetSQLPreview	Integer	Sets the current SQL statement for the DataWindow control.
SetSQLSelect	Integer	Changes the current SELECT statement for the DataWindow control.
SetTabOrder	Integer	Changes the tab value of the specified column in the DataWindow control.
SetText	Integer	Replaces the text in the edit control at the current row and column of the DataWindow control with the specified text.
SetTrans	Integer	Sets values in the DataWindow control's internal Transaction object.
SetTransObject	Integer	Sets the Transaction object for the DataWindow control and provides control over the transaction, including the ability to commit from a script.
SetValidate	Integer	Changes the validation rule used for the specified column of the DataWindow control.
SetValue	Integer	Sets the value of the specified item in the value list or the code table of the specified column of the DataWindow control.

DataWindow function	Datatype returned	Description
SetWSObject	Integer	Causes a DataWindow control to use a programmer-specified connection object. The connection object provides the information necessary for communicating with a Web service data source.
ShareData	Integer	Shares data between a primary DataWindow control and a secondary DataWindow control.
ShareDataOff	Integer	Turns off sharing for the DataWindow control. If that control is the primary DataWindow control, all secondary DataWindow controls are disconnected and their DataWindow objects no longer contain data.
Show	Integer	Makes the DataWindow control visible.
ShowHeadFoot	Integer	In a RichText presentation style DataWindow control, displays the panels for editing the header and footer or hides the panels and returns to editing the main text.
Sort	Integer	Sorts the rows of the DataWindow control based on its current sort criteria.
TextLine	String	Reports information about the edit control over the current row and column.
TriggerEvent	Integer	Triggers a specified event in the DataWindow control and executes the script for the event.
TypeOf	Object	Returns the type of the control.
Undo	Integer	Cancels the last edit in the edit control over the current row and column.
Update	Integer	Sends to the database all inserts, deletes, and updates of the DataWindow control.

DataWindowChild object

A DataWindowChild object is a nested report or a DropDownDataWindow within a DataWindow object. For example, a DataWindow object that populates a column having the DropDownDataWindow edit style is a DataWindowChild object.

The DataWindowChild object is used for accessing DataWindow objects independently from DataWindow functionality, and it inherits from the system Structure object because it needs storage and autoinstantiation.

A DataWindowChild object has no events.

Obsolete functions
Several DataWindowChild functions are described as obsolete, which means that although the functions operate as usual in this release, they will be removed in a future release. You should replace all use of these functions as soon as possible.

Properties

DataWindowChild property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Functions

DataWindowChild function	Datatype returned	Description
AcceptText	Integer	Applies the contents of the edit control to the current item in the DataWindowChild buffer.
ClassName	String	Returns the name assigned to the DataWindowChild.
ClearValues	String	Deletes all the items from a value list or code table associated with a DataWindow column.
CrosstabDialog	Integer	Displays the Crosstab Definition dialog box so that the user can modify the definition of a crosstab DataWindow object during execution.
DBCancel	Integer	Cancels a database retrieval in progress.
DBErrorCode	Long	Returns the error code (number) generated by a database error. **Obsolete function** DBErrorCode is an obsolete function and will be discontinued in a future release. Database error codes are now available as event arguments.
DBErrorMessage	String	Returns a string containing the text of the error message generated by a database error. **Obsolete function** DBErrorMessage is an obsolete function and will be discontinued in a future release. Database error messages are now available as event arguments.

DataWindowChild function	Datatype returned	Description
DeletedCount	Long	Returns the number of rows that have been deleted from the DataWindowChild but have not yet been updated in the associated database table.
DeleteRow	Integer	Deletes the specified row from the DataWindowChild.
Describe	String	Returns requested information about the structure of the DataWindowChild.
Filter	Integer	Displays specific rows of the DataWindowChild based on its current filter.
FilteredCount	Integer	Returns the number of rows that are not visible because of the DataWindowChild's current filter.
Find	Long	Returns the number of the first row that meets the search criteria within a specified search range in the detail area of a DataWindowChild.
FindGroupChange	Long	Searches starting at a specified row for the first break for the specified group in the DataWindowChild.
GetBandAtPointer	String	Returns the string containing the band in which the pointer is currently located followed by a tab character (~t) and the number of the row associated with the band.
GetBorderStyle	Border (enumerated)	Returns a Border enumerated datatype indicating the border style of the specified column in the DataWindowChild. Values are: Box! Lowered! NoBorder! Raised! ResizeBorder! ShadowBox! Underline!
GetChanges	Long	Retrieves changes made to a DataWindow into a blob. This function is used primarily in distributed applications.
GetChild	Integer	Provides a reference to a child DataWindow or to a report in a composite DataWindow, which you can use in DataWindow functions to manipulate that DataWindow or report.
GetClickedColumn	Integer	Returns the number of the column in the DataWindowChild that the user clicked or double-clicked.
GetClickedRow	Long	Returns the number of the row in the DataWindowChild that the user clicked or double-clicked.
GetColumn	Integer	Returns the number of the current column in the DataWindowChild.
GetColumnName	String	Returns the name of the current column in the DataWindowChild.

DataWindowChild function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetFormat	String	Returns the format used for display in the specified column of the DataWindowChild.
GetItemDate	Date	Returns the date data in the specified row and column of the DataWindowChild.
GetItemDateTime	DateTime	Returns the datetime data in the specified row and column of the DataWindowChild.
GetItemDecimal	Decimal	Returns the decimal data in the specified row and column of the DataWindowChild.
GetItemNumber	Double	Returns the numeric data in the specified row and column of the DataWindowChild.
GetItemStatus	dwItemStatus (enumerated)	Returns the status of the item at the specified row and column location in the specified buffer. Values are: DataModified! New! NewModified! NotModified!
GetItemString	String	Returns the string data in the specified row and column of the DataWindowChild.
GetItemTime	Time	Returns the time data in the specified row and column of the DataWindowChild.
GetNextModified	Long	Returns the number of the first row that was modified in the specified buffer in the specified DataWindowChild after the specified row.
GetObjectAtPointer	String	Returns the string containing the name of the DataWindowChild column or graphic control under the pointer in the DataWindowChild, then a tab character (~t), and then the row number.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetRow	Long	Returns an integer containing the number of the current row in the DataWindowChild.
GetRowFromRowId	Long	Gets the row number of a row in a DataWindow control from the unique row identifier associated with that row.
GetRowIdFromRow	Long	Gets the unique row identifier of a row in a DataWindow control from the row number associated with that row.
GetSelectedRow	Integer	Returns the number of the first selected row after the specified row number in the DataWindowChild.

DataWindowChild function	Datatype returned	Description
GetSQLPreview	String	Returns the current SQL statement the DataWindowChild is submitting to the database. **Obsolete function** GetSQLPreview is an obsolete function and will be discontinued in a future release. SQL syntax is now available as an event argument.
GetSQLSelect	String	Returns the current SELECT statement for the DataWindowChild.
GetText	String	Returns the text in the edit control over the current row and column of the DataWindowChild.
GetTrans	Integer	Returns the values in the DataWindowChild Transaction object.
GetUpdateStatus	Integer	Stores the number of the row that will be updated in a variable and the dwBuffer enumerated datatype identifying the buffer containing the row in another variable. **Obsolete function** GetUpdateStatus is an obsolete function and will be discontinued in a future release. Update status is now available as an argument in the DataWindow DBError and SQLPreview events.
GetValidate	String	Returns the validation rule used in the specified column of the DataWindowChild.
GetValue	String	Returns the specified item in the value list for the specified column.
GroupCalc	Integer	Recalculates the breaks in the groups in the DataWindowChild.
ImportClipboard	Long	Copies data from the clipboard to the DataWindowChild.
ImportFile	Long	Copies data from a file to the DataWindowChild.
ImportString	Long	Copies data from a string to the DataWindowChild.
InsertRow	Long	Inserts a new initialized row before the specified row in the DataWindowChild.
IsSelected	Boolean	Returns true if the specified row in the DataWindowChild is selected; returns false if the row is not selected or is greater than the number of rows in the DataWindowChild.
ModifiedCount	Long	Returns the number of rows that have been modified in the DataWindowChild but have not yet been updated in the associated database table.
Modify	String	Uses the specification contained in a string to modify the DataWindowChild.
OLEActivate	Integer	Activates OLE for the OLE object in the specified row and column of the DataWindowChild.
ReselectRow	Integer	Accesses the database to reselect all columns that can be updated and refreshes all timestamp columns in a row in the DataWindowChild.

DataWindowChild function	Datatype returned	Description
Reset	Integer	Clears all the data from a DataWindowChild.
ResetTransObject	Integer	Stops the DataWindowChild from using a programmer-defined Transaction object (thereafter, the DataWindow uses its internal Transaction object).
ResetUpdate	Integer	Resets the update flags for the DataWindowChild.
Retrieve	Long	Causes the DataWindowChild to retrieve rows from the database.
RowCount	Long	Returns the number of rows currently available in the DataWindowChild (all the rows retrieved minus any deleted rows plus any inserted rows minus any rows that have been filtered out).
RowsCopy	Integer	Copies a range of rows from one DataWindowChild to another or from one buffer to another within a single DataWindowChild.
RowsDiscard	Integer	Discards a range of rows. The rows cannot be restored unless retrieved from the database.
RowsMove	Integer	Clears a range of rows from a DataWindowChild and inserts the rows in another DataWindowChild or another buffer of the same DataWindowChild.
SaveAs	Integer	Saves the contents of the DataWindowChild control to the specified file, in the specified format, with or without column headings at the beginning
ScrollNextPage	Long	Scrolls forward by the number of rows showing in the DataWindowChild.
ScrollNextRow	Long	Scrolls the DataWindowChild to the next row. ScrollNextRow changes the current row but does not change the current column.
ScrollPriorPage	Long	Scrolls backward by the number of rows showing in the DataWindowChild.
ScrollPriorRow	Long	Scrolls to the previous row. The ScrollPriorRow function changes the current row in the DataWindowChild but does not change the current column.
ScrollToRow	Integer	Causes the control to scroll to the specified row. ScrollToRow changes the current row in the DataWindowChild but does not change the current column.
SelectRow	Integer	Selects or deselects the specified row of the DataWindowChild.
SetBorderStyle	Integer	Sets the border style of the specified column in the DataWindowChild.
SetChanges	Long	Applies changes captured with GetChanges to a DataWindow. This function is used primarily in distributed applications.
SetColumn	Integer	Makes the specified column the current column in the DataWindowChild.

DataWindowChild function	Datatype returned	Description
SetDetailHeight	Integer	Sets the height of each row in a specified range.
SetFilter	Integer	Defines the filter criteria for the DataWindowChild. The actual filtering is performed by the Filter function.
SetFormat	Integer	Sets the display format for the specified column of the DataWindowChild.
SetItem	Integer	Sets the value of the specified row and column of the specified DataWindowChild.
SetItemStatus	Integer	Sets the status of a row in a specified column of the DataWindowChild in the specified buffer.
SetPosition	Integer	Moves an object within the DataWindowChild to another band or changes the front-to-back order of objects within a band.
SetRedraw	Integer	Controls automatic redrawing of the DataWindowChild after each change in its properties or contents.
SetRow	Integer	Makes the specified row the current row in the DataWindowChild.
SetRowFocusIndicator	Integer	Sets the current row indicator for the DataWindowChild.
SetSort	Integer	Defines the sort criteria for the DataWindowChild. The actual sorting is performed by the Sort function.
SetSQLPreview	Integer	Sets the current SQL statement for the DataWindowChild.
SetSQLSelect	Integer	Changes the current SELECT statement for the DataWindowChild.
SetTabOrder	Integer	Changes the tab value of the specified column in the DataWindowChild.
SetText	Integer	Replaces the text in the edit control at the current row and column of the DataWindowChild with the specified text.
SetTrans	Integer	Sets values in the DataWindowChild's internal Transaction object.
SetTransObject	Integer	Sets the Transaction object for the DataWindowChild and provides control over the transaction, including the ability to commit from a script.
SetValidate	Integer	Changes the validation rule used for the specified column of the DataWindowChild.
SetValue	Integer	Sets the value of the specified item in the value list or the code table of the specified column of the DataWindowChild.
SetWSObject	Integer	Causes a DataWindowChild to use a programmer-specified connection object. The connection object provides the information necessary for communicating with a Web service data source.

DataWindowChild function	Datatype returned	Description
ShareData	Integer	Shares data between a primary DataWindowChild and a secondary DataWindowChild.
ShareDataOff	Integer	Turns off sharing for the DataWindowChild. If that object is the primary DataWindowChild, all secondary DataWindowChild objects are disconnected and their DataWindow objects no longer contain data.
Sort	Integer	Sorts the rows of the DataWindowChild based on its current sort criteria.
TypeOf	Object	Returns the type of the control.
Update	Integer	Sends to the database all inserts, deletes, and updates of the DataWindowChild.

DatePicker control

A DatePicker control makes it easy for users to select a date. It has two parts: a drop-down list box that displays the date in a selected format, and a grid that resembles the MonthCalendar control. Unlike the MonthCalendar control, which can be used to select a range of dates, the DatePicker control is used to select a single date.

Properties

DatePicker property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	Describes the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes the kind of user interface element that the control is.

DatePicker property	Datatype	Description
AllowEdit	Boolean	Specifies whether the user can select the text string in the control and edit it. Values are: TRUE – The user can select the text string in the control and edit it. FALSE – The user can change the date only by modifying one part of the date at a time or by selecting a date from the drop-down calendar (default).
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border (default). FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
CalendarBackColor	Long	Specifies the numeric value of the background color of the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. The default is Window Background. This property does not work on the Windows 7/8.1/10Windows 7/8.1/10 operating system.
CalendarFontCharset	FontCharSet (enumerated)	Specifies the font character set used for the text in the calendar. The default is ansi!. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the PowerBuilder Browser. This property cannot be set in the painter.
CalendarFontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the calendar. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss! To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view.

DatePicker property	Datatype	Description
CalendarFontName	String	Specifies the name of the typeface in which the text in the calendar displays (for example, Arial or Tahoma). To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view. This property does not work on the Windows 7/8.1/10 operating system.
CalendarFontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the calendar. Values are: Default! Fixed! Variable! This property cannot be set in the painter.
CalendarFontWeight	Integer	Specifies the weight of the font in the calendar in the range 0 through 1000, where 400 is normal and 700 is bold. A default weight is used if FontWeight is 0. The default is normal. To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view. This property does not work on the Windows 7/8.1/10 operating system.
CalendarItalic	Boolean	Specifies whether the text in the calendar is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic (default). For more information, see Italic. To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTextColor	Long	Specifies the numeric value of the text color in the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. The default is Window Text. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTextSize	Integer	Specifies the size of text in the calendar. The default is 9. To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view.

DatePicker property	Datatype	Description
CalendarTitleBackColor	Long	Specifies the numeric value of the background color of the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTitleTextColor	Long	Specifies the numeric value of the color used for text in the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTrailingTextColor	Long	Specifies the numeric value of the color used for leading and trailing days in the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
CalendarUnderline	Boolean	Specifies that text in the calendar is underlined. The default is false. To set this property in the painter, select the Browse button next to the FontName field on the Calendar page in the Properties view. This property does not work on the Windows 7/8.1/10 operating system.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CustomFormat	String	Specifies a custom format for the display of the date in a DatePicker control.
DateValue	Date	Gets the date value assigned to the control. The default is the current date. This property cannot be set in the painter and should not be set in script. Use the Value property to set the date value.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to put the control into Drag mode manually by using the Drag function.

DatePicker property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
DropDownRight	Boolean	Specifies whether the drop-down calendar is aligned with the right or left side of the DatePicker control. Values are: TRUE – The calendar is aligned with the right side of the control. FALSE – The calendar is aligned with the left side of the control (default).
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected (default). FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Tahoma).
FirstDayOfWeek	WeekDay (enumerated)	Specifies which day of the week displays on the left in the calendar.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The default is ansi!. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
FontWeight	Integer	Specifies the weight of the font in the control in the range 0 through 1000, where 400 is normal and 700 is bold. A default weight is used if FontWeight is 0.

DatePicker property	Datatype	Description
Format	DateTimeValue (enumerated)	Specifies the format of the date displayed in a DatePicker control. Values are: DtfCustom! DtfLongDate! DtfShortDate! DtfTime!
Height	Integer	Specifies the height of the control in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic (default).
MaxDate	Date	Specifies the latest date the user can select from the calendar. The default is December 31, 2999.
MinDate	Date	Specifies the earliest date the user can select from the calendar. The default is January 1, 1800.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order.
ShowUpDown	Boolean	Specifies whether the control uses an up-down control to change the date and/or time. Values are: TRUE – The control has an up-down control. FALSE – The control has a drop-down arrow that displays a calendar (default). This property cannot be changed at runtime.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Gets the text associated with the control. The string returned is equivalent to the Value property with the Format or CustomFormat applied. This property cannot be set in the painter and should not be set in script.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
TimeValue	Time	Gets the time value assigned to the control. The default is the current time. This property cannot be set in the painter and should not be set in script. Use the Value property to set the time value.

DatePicker property	Datatype	Description
TodayCircle	Boolean	Specifies whether the border of today's date on the calendar displays in red. Values are: TRUE – The Today circle is displayed (default). FALSE – The Today circle is not displayed.
TodaySection	Boolean	Specifies whether the label "Today:" followed by the current date displays at the bottom of the calendar. Values are: TRUE – The Today section is displayed (default). FALSE – The Today section is not displayed. This property does not work correctly on the Windows 7/8.1/10 operating system.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined (default).
Value	DateTime	Specifies the date/time value assigned to the control. The default is the current date and time.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible (default). FALSE – Control is not visible.
WeekNumbers	Boolean	Specifies whether a number representing the number of the week in the year displays to the left of each row in the calendar. Values are: TRUE – Week numbers are displayed. FALSE – Week numbers are not displayed (default). This property does not work correctly on the Windows 7/8.1/10 operating system.
Width	Integer	Specifies the width of the control in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window) in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window) in PowerBuilder units.

Events

DatePicker event	Occurs
Clicked	When the control is clicked (selected) with the left mouse button
CloseUp	When the user has selected a date from the drop-down calendar and the calendar closes

DatePicker event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is clicked twice with the left mouse button
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
DropDown	When the user has clicked the drop-down arrow in a DatePicker control before the drop-down calendar displays
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Controls message occurs that is not a PowerBuilder event
PreCreateWindow	This event is reserved for future use
RButtonDown	When the right mouse button is pressed on the control
UserString	When the user has edited the contents of the control and the control has lost focus
ValueChanged	When the Value property in a DatePicker control changes

Functions

DatePicker function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetCalendar	Long	This function is reserved for future use
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
GetText	String	Returns the text displayed in the control
GetToday	Date	Returns the date that the calendar uses as today's date
GetValue	Integer	Returns the date and time in the Value property
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control

DatePicker function	Datatype returned	Description
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds the specified event to the end of the event queue for the specified object
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets focus to the specified control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
SetToday	Integer	Sets the value that is used by the calendar as today's date
SetValue	Integer	Sets the date and time in the Value property
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

DropDownListBox control

A DropDownListBox control combines the features of a ListBox and a SingleLineEdit. In some DropDownListBoxes, the user can select an item by entering the name of the item in the text box. In other DropDownListBoxes, the user cannot modify the text box and must click the item or enter the first character of the item to select it.

Making the list display

In the development environment, if the list portion of the DropDownListBox is not displayed because ShowList is set to FALSE, the user must click the down arrow at the end of the text box to display it.

Properties

DropDownList Box property	Datatype	Description
Accelerator	Integer	The ASCII value of the accelerator key you want to assign as the accelerator for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AllowEdit	Boolean	Specifies whether the user can enter text in the text box portion of the control. Values are: TRUE – Can enter text in the text box. FALSE – Cannot enter text in the text box. AllowEdit must be true when ShowList is true.
AutoHScroll	Boolean	Specifies whether the text box portion of the control scrolls horizontally automatically when data is entered or deleted. Values are: TRUE – TextBox scrolls horizontally automatically. FALSE – TextBox does not scroll horizontally automatically.
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. However, setting this property to FALSE has no effect on the DropDownListBox control.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder will move the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

DropDownList Box property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically put in Drag mode. FALSE – When the control is clicked, the control is not automatically put in Drag mode. You have to put the control into Drag mode manually by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.

DropDownList Box property	Datatype	Description
HScrollBar	Boolean	Specifies whether a horizontal scroll bar is displayed in the control. Values are: TRUE – Horizontal scroll bar is displayed FALSE – Horizontal scroll bar is not displayed
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic FALSE – Text is not italic
Item[ ]	String array	Specifies the contents of the ListBox portion of the DropDownListBox.
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) the user can enter in the SingleLineEdit portion of the DropDownListBox (0 means unlimited).
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer that will be used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order FALSE – Characters display in left-to-right order
ShowList	Boolean	Specifies whether the option list always displays in the ListBox portion of the DropDownListBox when the control displays. Values are: TRUE – Option list always displays. FALSE – Option list displays only when the user clicks the down arrow. This property is usually set to false. Note that AllowEdit must be true when ShowList is true.
Sorted	Boolean	Specifies whether the ListBox portion of the DropDownListBox is automatically sorted in ascending order. Values are: TRUE – ListBox automatically sorted. FALSE – ListBox not sorted.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text in the control.

DropDownList Box property	Datatype	Description
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed in the control. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

DropDownListBox event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)

DropDownListBox event	Occurs
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Modified	When the control loses focus, the text has been changed, and Enter or Tab is pressed.
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control
SelectionChanged	When an item is selected in the ListBox portion of the DropDownListBox

Functions

DropDownListBox function	Datatype returned	Description
AddItem	Integer	Adds a new item to the end of the ListBox portion of the control. The AddItem function does not update the Item[ ] property of this control.
ClassName	String	Returns the name assigned to the control.
Clear	Integer	Clears the selected text from the control (but does not place it in the clipboard).
Copy	Integer	Copies (but does not delete) the selected text from the control to the clipboard.
Cut	Integer	Cuts (deletes) the selected text (if any) from the control and places it in the clipboard.
DeleteItem	Integer	Deletes the item indicated by the index from the ListBox portion of the control.
DirList	Boolean	Populates the ListBox portion of the DropDownListBox with a list of the files of the specified type that match the specified file pattern.
DirSelect	Boolean	Retrieves the current selection from the specified control and puts it in the specified variable.
Drag	Integer	Starts or ends the dragging of the control.
FindItem	Integer	Finds the first item in the ListBox portion of the control (after the specified index) that begins with a specified string.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible (hidden).

DropDownListBox function	Datatype returned	Description
InsertItem	Integer	Adds a new item to the ListBox portion of the DropDownListBox before the item indicated by the index.
Move	Integer	Moves the control to a specified location.
Paste	Integer	Inserts the contents of the clipboard (if any) at the cursor location in the control.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
Position	Integer	Returns the position of the cursor in the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
ReplaceText	Integer	Replaces the selected text in the control with the specified string.
Reset	Integer	Deletes all items from the control.
Resize	Integer	Changes the size of the control.
SelectedLength	Integer	Returns the length of the selected text in the control.
SelectedStart	Integer	Returns the starting position of the selected text (if any) in the control.
SelectedText	String	Returns a string containing the selected text (if any) from the control (the AllowEdit property must be true).
SelectItem	Integer	Finds and highlights an item in the control. Use Syntax 1 when you know the text of the item but not its position. Use Syntax 2 when you know the position of the item in the control's list or you want to clear the current selection.
SelectText	Integer	Selects the text in the control specified by the starting position and length; when the control has focus, highlights the text.
SetFocus	Integer	Sets focus in the first item in the box.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
Text	String	Returns the text of the item in the ListBox portion of the DropDownListBox that is identified by the index.
TotalItems	Integer	Returns the total number of items in the ListBox portion of the DropDownListBox.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Control	Returns the type of the control.

DropDownPictureListBox control

A DropDownPictureListBox control is similar to a DropDownListBox, but with the addition of pictures associated with the items in the list.

The pictures used in this control can be bitmaps (*BMP* file), icons (*ICO* file), cursors (*CUR* file), *GIF* (but not animated *GIF*) files, or *JPEG* files.

Making the list display
In the development environment, if the list portion of the DropDownPictureListBox is not displayed because ShowList is set to FALSE, the user must click the down arrow at the end of the text box to display it.

Properties

DropDownPicture ListBox property	Datatype	Description
Accelerator	Integer	The ASCII value of the accelerator key you want to assign as the accelerator for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AllowEdit	Boolean	Specifies whether the user can enter text in the text box portion of the control. Values are: TRUE – Can enter text in the text box. FALSE – Cannot enter text in the text box. AllowEdit must be true when ShowList is true.
AutoHScroll	Boolean	Specifies whether the text box portion of the control scrolls horizontally automatically when data is entered or deleted. Values are: TRUE – Text box scrolls horizontally automatically. FALSE – Text box does not scroll horizontally automatically.
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.

DropDownPicture ListBox property	Datatype	Description
Border	Boolean	Specifies whether the control has a border. Note that setting this property to FALSE has no effect on the control; it always has a border. Values are: TRUE – Control has a border. FALSE – Not applicable.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder will move the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag Mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag Mode. FALSE – When the control is clicked, the control is not automatically in Drag Mode. You have to manually put the control into Drag Mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).

DropDownPicture ListBox property	Datatype	Description
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar is displayed in the control. Values are: TRUE – Horizontal scroll bar is displayed. FALSE – Horizontal scroll bar is not displayed.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Item[ ]	String array	Specifies the initial item strings in the ListBox portion of the DropDownPictureListBox. This array is not updated after initialization.
ItemPictureIndex[ ]	Integer	Initial picture index for each item in the Item property array. These values are not updated after initialization.
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) the user can enter in the SingleLineEdit portion of the DropDownPictureListBox (0 means unlimited).

DropDownPicture ListBox property	Datatype	Description
PictureHeight	Integer	Specifies the height of the picture, in pixels. This property can be set only when there are no images in the image list. If the value is 0 at the time the first image is added, the size of that image is used to set the size of the rest of the images added.
PictureMaskColor	Long	Specifies the numeric value of the color to be used to mask user-defined bitmaps added through the initial picture array or with the AddPicture function. System-defined bitmaps know their mask color and this color is ignored. This value is used when a picture is added, and therefore can be changed between AddPicture calls. Values can be: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
PictureName[ ]	String	Specifies the names of the files containing the pictures added during initialization. The file extension *BMP*, *ICO*, *GIF*, *JPG*, or *JPEG* is required. This property is not updated after initialization.
PictureWidth	Integer	Specifies the width of the picture, in pixels. This property can be set only when there are no images in the image list. If the value is 0 at the time the first image is added, the size of that image is used to set the size of the rest of the images added.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer that will be used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
ShowList	Boolean	Specifies whether the option list always displays in the ListBox portion of the DropDownPictureListBox when the control displays. Values are: TRUE – Option list always displays. FALSE – Option list displays only when the user clicks the down arrow. This property is usually set to false. AllowEdit must be true when ShowList is true.

DropDownPicture ListBox property	Datatype	Description
Sorted	Boolean	Specifies whether the ListBox portion of the DropDownPictureListBox is automatically sorted in ascending order. Values are: TRUE – ListBox automatically sorted. FALSE – ListBox not sorted.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text in the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed in the control. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

DropDownPicture ListBox event	Occurs
Constructor	Immediately before the Open event occurs in the window.
Destructor	Immediately after the Close event occurs in the window.
DoubleClicked	When the control is double-clicked (selected and activated).
DragDrop	When a dragged control is dropped on the control.
DragEnter	When a dragged control enters the control.
DragLeave	When a dragged control leaves the control.
DragWithin	When a dragged control is within the control.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
LoseFocus	When the control loses focus (becomes inactive).
Modified	When the control loses focus, the text has been changed, and enter or tab is pressed.
Other	When a Windows message occurs that is not a PowerBuilder event.
RButtonDown	When the right mouse button is pressed on the control.
SelectionChanged	When an item is selected in the ListBox portion of the DropDownPictureListBox.

Functions

DropDownPicture ListBox function	Datatype returned	Description
AddItem	Integer	Adds a new item to the end of the ListBox portion of the control. The AddItem function does not update the Item[ ] or ItemPictureIndex[ ] properties of this control.
AddPicture	Integer	Adds the bitmap, icon, or cursor file to the main image list. This function does not update the PictureName[ ] property.
ClassName	String	Returns the name assigned to the control.
Clear	Integer	Clears the selected text from the control (but does not place it in the clipboard).
Copy	Integer	Copies (but does not delete) the selected text from the control to the clipboard.
Cut	Integer	Cuts (deletes) the selected text (if any) from the control and places it in the clipboard.
DeleteItem	Integer	Deletes the item indicated by the index from the ListBox portion of the control.

DropDownPicture ListBox function	Datatype returned	Description
DeletePicture	Integer	Deletes the specified picture from the image list. This function does not update the PictureName[ ] property.
DeletePictures	Integer	Deletes all the pictures from the image list. This function does not update the PictureName[ ] property.
DirList	Boolean	Populates the ListBox portion of the DropDownPictureListBox with a list of the files of the specified type that match the specified file pattern.
DirSelect	Boolean	Retrieves the current selection from the specified control and puts it in the specified variable.
Drag	Integer	Starts or ends the dragging of the control.
FindItem	Integer	Finds the first item in the ListBox portion of the control (after the specified index) that begins with a specified string.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible (hidden).
InsertItem	Integer	Adds a new item to the ListBox portion of the DropDownPictureListBox before the item indicated by the index. This function does not update the Item[ ] or ItemPictureIndex[ ] properties of this control.
Move	Integer	Moves the control to a specified location.
Paste	Integer	Inserts the contents of the clipboard (if any) at the cursor location in the control.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
Position	Integer	Returns the position of the cursor in the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
ReplaceText	Integer	Replaces the selected text in the control with the specified string.
Reset	Integer	Deletes all items from the control.
Resize	Integer	Changes the size of the control.
SelectedLength	Integer	Returns the length of the selected text in the control.
SelectedStart	Integer	Returns the starting position of the selected text (if any) in the control.
SelectedText	String	Returns a string containing the selected text (if any) from the control (the AllowEdit property must be true).

DropDownPicture ListBox function	Datatype returned	Description
SelectItem	Integer	Finds and highlights an item in the control. Use Syntax 1 when you know the text of the item but not its position. Use Syntax 2 when you know the position of the item in the control's list or you want to clear the current selection.
SelectText	Integer	Selects the text in the control specified by the starting position and length; when the control has focus, highlights the text.
SetFocus	Integer	Sets focus in the first item in the box.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
Text	String	Returns the text of the item in the ListBox portion of the DropDownPictureListBox identified by the index.
TotalItems	Integer	Returns the total number of items in the ListBox portion of the DropDownPictureListBox.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Control	Returns the type of the control.

DynamicDescriptionArea object

DynamicDescriptionArea is a PowerBuilder system object that stores information about the input and output parameters used in Format 4 of dynamic SQL.

PowerBuilder provides a global DynamicDescriptionArea named SQLDA that you can use when you need a DynamicDescriptionArea variable. If necessary, you can declare and create additional variables of this type using this system object as the datatype.

For more information about using dynamic SQL, see the *PowerScript Reference*.

Properties

DynamicDescriptionArea property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
NumInputs	Integer	Specifies the number of input parameters found in the dynamic SQL PREPARE statement. PowerBuilder populates this property when the DESCRIBE statement is executed.
NumOutputs	Integer	Specifies the number of output parameters found in the PREPARE statement. If the database supports output parameter description, PowerBuilder populates this property when the DESCRIBE statement is executed. If the database does not support output parameter description, PowerBuilder populates this property when the FETCH statement is executed.
InParmType[ ]	ParmType (enumerated)	Array containing values specifying the datatype of each input parameter. Values are: TypeBoolean! TypeByte! TypeDate! TypeDateTime! TypeDecimal! TypeDouble! TypeInteger! TypeLong! TypeLongLong! TypeReal! TypeString! TypeTime! TypeUInt! TypeULong! TypeUnknown!
OutParmType[ ]	ParmType (enumerated)	Array containing values specifying the datatype of each output parameter returned.

Events

DynamicDescription Area event	Occurs
Constructor	Immediately before the Open event occurs
Destructor	Immediately after the Close event occurs

Functions

DynamicDescription Area function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetDynamicDate	Date	Obtains data of type Date from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicDate when the value of OutParmType is TypeDate! for the value in the array that you want to retrieve.
GetDynamicDateTime	DateTime	Obtains data of type DateTime from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicDateTime when the value of OutParmType is TypeDateTime! for the value in the array that you want to retrieve.
GetDynamicDecimal	LongLong	Obtains numeric data from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicDecimal when the value of OutParmType is TypeDecimal! or TypeLongLong! for the value in the array that you want to retrieve.
GetDynamicNumber	Double	Obtains numeric data from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicNumber when the value of OutParmType is TypeByte!, TypeInteger!, TypeDouble!, TypeLong!, TypeReal!, or TypeBoolean! for the value in the array that you want to retrieve.
GetDynamicString	String	Obtains data of type String from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicString when the value of OutParmType is TypeString! for the value in the array that you want to retrieve.
GetDynamicTime	Time	Obtains data of type Time from the DynamicDescriptionArea after you have executed a dynamic SQL statement. Use GetDynamicTime when the value of OutParmType is TypeTime! for the value in the array that you want to retrieve.

DynamicDescription Area function	Datatype returned	Description
GetParent	PowerObject	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds the specified event to the end of the message queue for the object.
SetDynamicParm	Integer	Specifies a value for an input parameter in the DynamicDescriptionArea that will be used in the SQL OPEN or EXECUTE statement. Use SetDynamicParm to fill the parameters in the input parameter descriptor array in the DynamicDescriptionArea.
TriggerEvent	Integer	Triggers a specified event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

DynamicStagingArea object

DynamicStagingArea is a PowerBuilder system object that stores information for use in dynamic SQL statements.

The DynamicStagingArea object is the only connection between the execution of a statement and a Transaction object and is used internally by PowerBuilder. You cannot access information in DynamicStagingArea, and there are no properties associated with DynamicStagingArea.

PowerBuilder provides a global DynamicStagingArea variable named SQLSA that you can use when you need a DynamicStagingArea variable. If necessary, you can declare and create additional variables of this type using this system object as the datatype.

For more information about using dynamic SQL, see the *PowerScript Reference*.

Properties

DynamicStagingArea property	DataType	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

DynamicStagingArea event	Occurs
Constructor	Immediately before the Open event occurs
Destructor	Immediately after the Close event occurs

Functions

DynamicStagingArea function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds the specified event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specified event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

EditMask control

An EditMask is a box similar to a SingleLineEdit in which the user can enter and edit one line of text. The type and number of characters entered is restricted by the edit mask, and the appearance of the text is specified by the edit mask. For example, you can use an EditMask to format a telephone number or date automatically as the user enters it.

Properties

EditMask property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the alignment of text in the control. Values are: Center! Justify! Left! Right!
AutoHScroll	Boolean	Specifies whether PowerBuilder automatically scrolls left or right when data is entered into or deleted from the control. Values are: TRUE – Scrolls horizontally automatically. FALSE – Does not scroll automatically.
AutoSkip	Boolean	Specifies whether to skip to the next control when the last character in the edit mask has been entered. Values are: TRUE – Skip to the next control automatically. FALSE – Do not skip to the next control.
AutoVScroll	Boolean	Specifies whether PowerBuilder automatically scrolls up or down when data is entered into or deleted from the control. Values are: TRUE – Scrolls vertically automatically. FALSE – Scrolling not automatic.

EditMask property	Datatype	Description
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window: TRUE – Control moved to top. FALSE – Control not moved to top.
CalendarBackColor	Long	Specifies the numeric value of the background color of the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. The default is Window Background. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTextColor	Long	Specifies the numeric value of the text color in the calendar:-2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. The default is Window Text. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTitleBackColor	Long	Specifies the numeric value of the background color of the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
CalendarTitleTextColor	Long	Specifies the numeric value of the color used for text in the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.

EditMask property	Datatype	Description
CalendarTrailingTextColor	Long	Specifies the numeric value of the color used for leading and trailing days in the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisplayData	String	Specifies the data that initially displays in the control.
DisplayOnly	Boolean	Specifies whether the text in the control is display only and cannot be changed by the user. Values are: TRUE – Text is display only. FALSE – Text can be changed by user.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
DropDownCalendar	Boolean	Whether a drop-down calendar displays in a control with a Date or DateTime edit mask. Values are: Yes – Drop-down calendar control displays. No – (Default) Drop-down calendar control does not display.
DropDownRight	Boolean	Specifies whether the drop-down calendar is aligned with the right or left side of the control. Values are: TRUE – The calendar is aligned with the right side of the control. FALSE – The calendar is aligned with the left side of the control (default).
Enabled	Boolean	Specifies whether the control is enabled (can be selected): TRUE – Control is enabled. FALSE – Control is not enabled.

EditMask property	Datatype	Description
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays in the control when all the data cannot be displayed at one time. Values are: TRUE – Horizontal scroll bar displayed. FALSE – Horizontal scroll bar not displayed.
HideSelection	Boolean	Specifies whether selected text stays selected (highlighted) even when the control does not have focus: TRUE – Text does not stay highlighted. FALSE – Text stays highlighted.
IgnoreDefaultButton	Boolean	Specifies whether the Clicked event for the window's Default command button is triggered when user presses Enter. Values are: TRUE – Do not trigger Clicked event; add new line in control. FALSE – Trigger Clicked event; do not add new line in control (default).
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Increment	Double	Specifies the increment of the spin.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.

EditMask property	Datatype	Description
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) that can be entered in the control (0 means unlimited).
Mask	String	Specifies the mask to use to format and edit data in the control.
MaskDataType	MaskDataType (enumerated)	Specifies the datatype of the control. Values are: DateMask! DateTimeMask! DecimalMask! NumericMask! StringMask! TimeMask!
MinBox	String	Specifies the minimum and maximum values for the spin.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
Spin	Boolean	Specifies whether to scroll through the spin values. Values are: TRUE – Scroll through the spin values. FALSE – Do not scroll through the spin values.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
TabStop[ ]	Integer	Specifies the positions of tab stops in the control.
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextColor	Long	Specifies the color to be used for the text in the control. For more information about color, see the RGB function in the *PowerScript Reference*.
TextCase	TextCase (enumerated)	Specifies the case used to display text the user enters. Values are: AnyCase! Lower! Upper!
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10. For information about TextSize and EditMask behavior, see online Help.

EditMask property	Datatype	Description
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
UseCodeTable	Boolean	Specifies whether PowerBuilder uses the code table for the column to validate data. Values are: TRUE – Uses code table to validate data. FALSE – Does not use code table to validate data.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar displays in the control when not all the data can be displayed at one time. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

EditMask event	Occurs
Constructor	Immediately before the Open event occurs in the window.
Destructor	Immediately after the Close event occurs in the window.
DragDrop	When a dragged control is dropped on the control.
DragEnter	When a dragged control enters a target control.
DragLeave	When a dragged control leaves the control.
DragWithin	When a dragged control is within the control.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
LoseFocus	When the control loses focus (becomes inactive).
Modified	When a control has been changed and loses focus (becomes inactive).

EditMask event	Occurs
Other	When a Windows message occurs that is not a PowerBuilder event.
RButtonDown	When the right mouse button is pressed on the control.

Functions

EditMask function	Datatype returned	Description
CanUndo	Boolean	Returns true if the Undo function can be used to undo the last edit in the control and returns false if it cannot.
ClassName	String	Returns the name assigned to the control.
Clear	Integer	Clears the selected text (if any) from the control but does not place it in the clipboard.
Copy	Integer	Copies (but does not delete) the selected text (if any) from the control to the clipboard.
Cut	Integer	Cuts (deletes) the selected text (if any) from the control and places it in the clipboard.
Drag	Integer	Starts or ends the dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetData	Integer	Obtains the unformatted data in the control.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
LineCount	Integer	Returns the number of lines in the EditMask in the window.
LineLength	Integer	Returns the length of the line in which the cursor is positioned.
Move	Integer	Moves the control to the specified location.
Paste	Integer	Inserts the contents of the clipboard at the insertion point in the specified control.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
Position	Integer	Returns the position of the insertion point in the control.
PostEvent	Boolean	Adds the specified event to the end of the message queue for the control.
Print	Integer	Prints the control.
ReplaceText	Integer	Replaces the currently selected text with the specified string. If no text is selected, inserts the text at the insertion point.
Resize	Integer	Changes the width and height of the control.

EditMask function	Datatype returned	Description
Scroll	Integer	Moves the contents of the control up or down the specified number of lines.
SelectedLength	Integer	Returns the total number of characters and spaces (length) in the selected text in the control.
SelectedLine	Integer	Returns the number of the line where the insertion point is located in the control.
SelectedStart	Integer	Returns the position of the first character in the selected text in the control.
SelectedText	String	Determines what if any text is selected in the control.
SelectText	Integer	Selects text in the control beginning at the specified position and continuing for the specified number of characters.
SetFocus	Integer	Sets focus to the control.
SetMask	Integer	Specifies the contents and datatype of the edit mask for the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TextLine	String	Returns the entire text of the line in which the insertion point is currently located.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.
Undo	Integer	Cancels the previous editing function performed in the control.

EnumerationDefinition object

Information about the type of a variable when it is an enumerated datatype. EnumerationDefinition is inherited from TypeDefinition. It has no events.

Properties

Enumeration Definition property	Datatype	Description
Category	TypeCategory	Specifies whether the type is simple, enumerated, or a class or structure. Values are: SimpleType! EnumeratedType! ClassOrStructureType!
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DataTypeOf	String	The system class name or simple datatype of the variable. DataTypeOf is a string representation of a value of the Object enumerated datatype. Values are lowercase with no exclamation point. Sample values include: window string any dropdownlistbox For objects you have defined, the datatype is the system class from which your object is inherited.
Enumeration[ ]	EnumerationItem Definition	An array of the name-value pairs for all the items in the enumeration.
IsStructure	Boolean	Indicates whether the type is a structure. Always FALSE.
IsSystemType	Boolean	Indicates whether the class is a system class—that is, one of the classes defined within PowerBuilder as opposed to a class defined in a PBL by a user.
IsVariableLength	Boolean	Specifies whether the datatype has a fixed size. Always true. Values are: TRUE – The datatype is of variable length, meaning the datatype is a string, any, blob, or unbounded array. FALSE – The datatype is a fixed length.
IsVisualType	Boolean	Indicates whether the type is a visual or nonvisual type. Always false.
LibraryName	String	The fully qualified name of the library the class was loaded from. The library can no longer contain the class. If a program manipulates the contents of libraries, the class could have been moved or deleted after it was loaded.
Name	String	The name of the class. For a nested class, the name is returned in the form of *libraryEntryName`className*.

Functions

Enumeration Definition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

EnumerationItemDefinition object

A class that provides information about the value names and the associated numeric values for an enumerated datatype. It is used in the EnumerationDefinition class. It has no events.

Properties

Enumeration ItemDefinition property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Name	String	The name of an enumerated value. For example, Left! is a named value of the enumerated datatype Alignment.
Value	Long	The numeric value associated with the name. For example, 0 is the value PowerBuilder associates with Left! The numeric value is important only if you are constructing source code for an object. Within PowerBuilder, you use the named value so that the datatype is correct.

Functions

Enumeration ItemDefinition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.

Enumeration ItemDefinition function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

Environment object

The Environment object is a system structure used to hold information about the computing platform the PowerBuilder application is running on. You populate the Environment object using the GetEnvironment function.

The Environment object has no events.

For more information about the GetEnvironment function, see the *PowerScript Reference.*

Properties

Environment property	Datatype	Description
CharSet	CharSet (enumerated)	The international character set used by PowerBuilder. Values include: • CharSetAnsi! • CharSetUnicode! • CharSetDBCS! • CharSetDBCSJapanese! The values CharSetAnsiArabic! and CharSetAnsiHebrew! are not valid choices in PowerBuilder 6 or later.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CPUType	CPUTypes (enumerated)	The type of CPU. For a complete list of CPUTypes values, see the Enumerated tab of the Browser.
Language	LanguageID (enumerated)	Specifies the value of the language setting for the machine. For a complete list of LanguageID values, see the Enumerated tab of the Browser.

Environment property	Datatype	Description
MachineCode	Boolean	Specifies whether the application executable is machine code (compiled). Values are: TRUE – Executable is machine code. FALSE – Executable is not machine code (pseudo-code).
OSFixesRevision	Integer	The maintenance version of the operating system.
OSMajorRevision	Integer	The major version of the operating system. For example, this value would be 4 for Windows 95, 98, ME, and NT 4.x, 5 for Windows 2000, XP, or 2003, and 6 for Vista, Windows Server 2008, and Windows 7.
OSMinorRevision	Integer	The point release of the operating system. For example, this value would be 0 for Windows Server 2008 and Vista, 1 for Windows XP or Windows 7, 2 for Windows Server 2003 and 64-bit XP, and 10 for SunOS 5.10 (Solaris 10).
PBBuildNumber	Integer	The build number of this version of PowerBuilder.
PBFixesRevision	Integer	The maintenance version of PowerBuilder.
PBMajorRevision	Integer	The major version of PowerBuilder.
PBMinorRevision	Integer	The point release of PowerBuilder.
NumberOfColors	LongLong	Number of colors on the screen.
ScreenHeight	Long	Height of the screen in pixels.
ScreenWidth	Long	Width of the screen in pixels.
OSType	OSTypes (enumerated)	Operating system or environment. For a complete list of OSType values, see the Enumerated tab of the Browser.
PBType	PBTypes (enumerated)	Version of the PowerBuilder product. For a complete list of PBType values, see the Enumerated tab of the Browser.
Win16 (obsolete)	Boolean	Indicates the type of the operating system in which the application executable is running. Values are: TRUE – Executable is running under a 16-bit operating system. FALSE – Executable is running under a 32-bit operating system.

Functions

Environment function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.

Environment function	Datatype returned	Description
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

Error object

The Error object is used to record execution-time errors. You can access the Error object from a script (typically in the SystemError event) to learn which error occurred and where it occurred. You can also customize your own version of the Error object by defining a class user object inherited from the built-in Error object.

For more information about creating a custom Error object, see the chapter on user objects in the *PowerBuilder Users Guide*. For information on using try-catch blocks to catch runtime and user-defined exceptions, see *Application Techniques* and the *PowerScript Reference*.

Properties

Error property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Line	Integer	Identifies the line in the script at which the error occurred.
Number	Integer	Identifies the PowerBuilder error.
Object	String	Contains the name of the object in which the error occurred. If the error occurred in a window or menu, Object will be the same as WindowMenu.
ObjectEvent	String	Contains the event in which the error occurred.
Text	String	Contains the text of the error message.
WindowMenu	String	Contains the name of the window or menu in which the error occurred.

Events

Error event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.

Functions

Error function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds an event to the end of the message queue of the user object.
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event.
TypeOf	Object	Returns the type of the user object.

ErrorLogging object

The ErrorLogging object provides the ability to write messages to the log file used by the object's container, such as the NT system application log for Microsoft Transaction Server.

Properties

ErrorLogging property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Events

ErrorLogging event	Occurs
Constructor	When the object is created.
Destructor	When the object is destroyed.

Functions

ErrorLogging function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Log	None	Writes a string to the log file maintained by the object's container.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

Exception object

The Exception object inherits from the Throwable object and is the base class for user-defined or "checked" exceptions. A function or user-defined event must declare each checked exception it throws, and a caller to a function or event that throws a checked exception must either catch the exception or throw the exception itself.

Unlike RuntimeError objects, Exception objects do not have built-in properties that provide information about the location where the error occurred.

The CORBAUserException system object inherits from Exception. This object maps to the CORBA_USER_EXCEPTION object that can be thrown from CORBA applications when user-defined exceptions have not been handled. Unlike the CORBASystemException object and its descendants, the CORBAUserException object is a checked exception and must be declared and caught like other checked exceptions.

Properties

Exception property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Text	String	Contains the text of the error message

Events

Exception event	Occurs
Constructor	Immediately before the exception is thrown
Destructor	Immediately after the exception is thrown

Functions

Exception function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetMessage	String	Returns the error message from objects of type Throwable.
GetParent	PowerObject	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
SetMessage	—	Sets an error message for an object of type Throwable.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

Graph object

A graph is a representation of a series of data points (values). The graph can have a single series of values or multiple series.

Properties

Graph property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window or DataWindow control.
Category	grAxis	Specifies the properties of the category axis of the graph.
CategorySort	grSortType	Specifies how the categories are sorted.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Depth	Integer	Specifies the percent the depth is of the width of the graph.
DragAuto	Boolean	Specifies whether PowerBuilder puts the graph automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags a control (the *ICO* file). The default icon is a box the size of the control. When the user drags a control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Elevation	Integer	Specifies the angle of front-to-back elevation.

Graph property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FocusRectangle	Boolean	Specifies whether a dotted rectangle (the focus rectangle) frames the control when it has focus. Values are: TRUE – Control will be framed when it has focus. FALSE – Control will not be framed when it has focus.
GraphType	grGraphType (enumerated)	Specifies the type of the graph. Values are: Area3D! AreaGraph! Bar3DGraph! Bar3DObjGraph! BarGraph! BarStack3DObjGraph! BarStackGraph! Col3DGraph! Col3DObjGraph! ColGraph! ColStack3DObjGraph! ColStackGraph! Line3D! LineGraph! Pie3D! PieGraph! ScatterGraph!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Legend	grLegendType (enumerated)	Specifies the alignment of the text in the graph legend. Values are: AtBottom! AtLeft! AtRight! AtTop! NoLegend!
LegendDispAttr	grDispAttr	Specifies the type style for the text in the graph legend, including the text style, size, color, and rotation.
OverlapPercent	Integer	Specifies the percent of the width of the data markers by which different series in a graph overlap.
Perspective	Integer	Specifies the distance the graph is from the front of the window.
PieDispAttr	grDispAttr	Specifies properties of the text in pie graph labels, including the text style, size, color, and rotation.

Graph property	Datatype	Description
Pointer	String	Contains the name of the stock pointer or the file containing the pointer used for the graph.
Render3D	Boolean	Indicates whether the 3D graph is rendered in the DirectX style.
Rotation	Integer	Specifies how much to rotate the graph from left to right.
Series	grAxis	Specifies the series in the graph.
SeriesSort	grSortType	Specifies how the series are sorted.
ShadeColor	Long	Specifies the color used for the shading in the graph.
Spacing	Integer	Specifies the space between data markers in the graph as a percent.
TabOrder	Integer	Specifies the tab value of the control in the tabbing sequence (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value of the control.
TextColor	Long	Specifies the color to be used for the text in the control.
Title	String	Specifies the text of the title for the graph.
TitleDispAttr	grDispAttr	Specifies the type style for the text in the graph title, including the text style, size, color, and rotation.
Values	grAxis	Specifies the values of the value axis of the graph.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the stroke weight of text in the control; for example, 400 for normal or 700 for bold.
X	Integer	Specifies the X position (the distance from the left edge of the parent window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the parent window), in PowerBuilder units.

Events

Graph event	Occurs
Clicked	When the control is clicked (selected or unselected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected or unselected)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When dragged control leaves the control

Graph event	Occurs
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

Graph function	Datatype returned	Description
AddCategory	Integer	Adds a category to the graph.
AddData	Long	Adds a value to the end of the specified series for the graph.
AddSeries	Integer	Adds a series to the graph and assigns the series a number.
CategoryCount	Integer	Counts the categories in the graph.
CategoryName	String	Obtains the name of the specified category in the graph.
ClassName	String	Returns the name assigned to the control.
Clipboard	Integer	Copies the graph in bitmap (BMP) format to the clipboard.
DataCount	Long	Returns the number of data points in a specified series in the graph.
DeleteCategory	Integer	Deletes the specified category and the data values in the series from the graph.
DeleteData	Integer	Deletes the data value in the specified data point in the specified series in the graph.
DeleteSeries	Integer	Deletes the specified series and the data values in the series from the graph.
Drag	Integer	Starts or ends dragging of the control.
FindCategory	Integer	Obtains the number of the specified category in the graph.
FindSeries	Integer	Obtains the number PowerBuilder assigned to the specified series when it was added to the graph.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetData	Double	Obtains the value of the specified data point in the specified series in the graph. See also the GetDataValue function.
GetDataLabelling	Integer	Determines whether the data at a given data point is labeled in a DirectX 3D graph.

Graph function	Datatype returned	Description
GetDataPieExplode	Integer	Reports the percentage by which a pie slice is exploded in a pie graph.
GetDataStyle	Integer	Finds out the appearance of a data point in a series. Each data point in a series can have individual appearance settings. There are different syntaxes, depending on what settings you want to check: colors, line style, fill pattern, or symbol.
GetDataTransparency	Integer	Obtains the transparency percentage of a series in a DirectX 3D graph.
GetDataValue	Integer	Returns the value of the specified data in the specified series in the specified graph.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetSeriesLabelling	Integer	Obtain the series labelling for a DirectX 3D graph.
GetSeriesStyle	Integer	Finds out the appearance of a series in a graph. The appearance settings for individual data points can override the series setting, so the values obtained from this function may not reflect the current state of the graph. There are several syntaxes, depending on what you want to get: colors; line style, fill pattern, or symbol; or whether the series is an overlay.
GetSeriesTransparency	Integer	Obtains the tranparency percentage of a series in a DirectX 3D graph.
Hide	Integer	Makes the control invisible.
ImportClipboard	Long	Copies the contents of the clipboard to the graph, starting in the specified column.
ImportFile	Long	Copies the contents of the specified file to the graph, starting in the specified column.
ImportString	Long	Imports the contents from the specified string to the graph, starting in the specified column.
InsertCategory	Integer	Inserts a new category before an existing category in the graph.
InsertData	Long	Inserts a new data point into the graph before a specified data point and moves existing data points to the right.
InsertSeries	Integer	Inserts a new series before an existing series in the graph.
ModifyData	Integer	Changes the value of the specified data point in the specified series in the graph to the specified value, and optionally modifies the data for the specified tick mark.
Move	Integer	Moves the control to a specified location.
ObjectAtPointer	GrObject Type	Stores the number of the series the pointer is over in the graph and the number of the specified data point in reference values, and identifies the object type.

Graph function	Datatype returned	Description
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Reset	Integer	Deletes data in the graph as specified.
ResetDataColors	Integer	Resets the color of a data point to the series color.
Resize	Integer	Changes the size of the control.
SaveAs	Integer	Saves the contents of the graph to a file in the specified format.
SeriesCount	Integer	Determines how many series there are in the graph.
SeriesName	String	Obtains the name of the specified series in the graph.
SetDataPieExplode	Integer	Explodes a pie slice in a pie graph.
SetDataLabelling	Integer	Set the data label for a DirectX 3D graph.
SetDataStyle	Integer	Specifies the appearance of a data point in a graph. There are several syntaxes, depending on what you want to set: colors, line style and width, or fill pattern or symbol.
SetDataTransparency	Integer	Sets the tranparency percentage of a data point in a series in a DirectX 3D.
SetFocus	Integer	Sets focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after a change in its properties.
SetSeriesLabelling	Integer	Set the series label for a DirectX 3D graph.
SetSeriesStyle	Integer	Specifies the appearance of a series in a graph. There are several syntaxes, depending on what you want to change: colors, line style and width, fill pattern or symbol, or whether the series is an overlay.
SetSeriesTransparency	Integer	Sets the transparency percentage of a series in a DirectX 3D graph.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

grAxis object

The PowerBuilder class grAxis is used as part of a graph control. Each graph has three grAxis objects: Category, Series, and Values.

To refer to a property of an axis, use this syntax:

graphcontrolname.axisname.property

For example: `gr_1.Series.AutoScale`

A grAxis object has no events.

Properties

grAxis property	Datatype	Description
AutoScale	Boolean	Specifies whether PowerBuilder scales the axis automatically. Values are: TRUE – Automatically scale. FALSE – Do not automatically scale.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DataType	grAxisDataType (enumerated)	Specifies the datatype of the axis. Values are: AdtDate! AdtDateTime! AdtDefault! AdtDouble! AdtText! AdtTime!
DispAttr	grDispAttr (object)	Settings that control the appearance of the text that labels the axis divisions.
DisplayEveryNLabels	Integer	Specifies which major axis divisions to label. For example, 2 means label every other tick mark. Use 0 to let the graph select the optimum number of labels to use. If the labels are too long, they are clipped.
DropLines	LineStyle (enumerated)	Specifies the type of drop line for the axis. Values are: Transparent! – No line appears. Continuous! – A solid line. Dash! Dot! DashDot! DashDotDot!

grAxis property	Datatype	Description
Frame	LineStyle (enumerated)	Specifies the type of line used for the frame. See DropLines in this table for the list of values.
Label	String	Specifies the axis label.
LabelDispAttr	grDispAttr (object)	Settings that control the appearance of the axis label text.
MajorDivisions	Integer	Specifies the number of major divisions on the axis.
MajorGridLine	LineStyle (enumerated)	Specifies the type of line for the major grid. See DropLines in this table for the list of values.
MajorTic	grTicType (enumerated)	Specifies the type of the major tick marks. Values are: NoTic! Inside! Outside! Straddle!
MaximumValue	Double	Specifies the maximum value for the axis when its datatype is numeric.
MaxValDateTime	DateTime	Specifies the maximum value for the axis when its datatype is date or time.
MinimumValue	Double	Specifies the minimum value for the axis when its datatype is numeric.
MinorDivisions	Integer	Specifies the number of minor divisions on the axis.
MinorGridLine	LineStyle (enumerated)	Specifies the type of line for the minor grid. See DropLines in this table for the list of values.
MinorTic	grTicType (enumerated)	Specifies the type of the minor tick marks. Values are: NoTic! Inside! Outside! Straddle!
MinValDateTime	DateTime	Specifies the minimum value for the axis when its datatype is date or time.
OriginLine	LineStyle (enumerated)	Specifies the type of origin line for the axis. See DropLines in this table for the list of values.
PrimaryLine	LineStyle (enumerated)	Specifies the type of primary line for the axis. See DropLines in this table for the list of values.
RoundTo	Double	Specifies the value to which you want to round the axis values.
RoundToUnit	grRoundToType (enumerated)	Specifies the units for the rounding value. The units must be appropriate for the axis datatype. Values are listed below. For an axis of any datatype: RndDefault! For an axis of any numeric datatype: RndNumber!

grAxis property	Datatype	Description
		For an axis of type date or DateTime: RndYears! RndMonths! RndDays! For an axis of type time or DateTime: RndHours! RndMinutes! RndSeconds! RndMicroseconds!
ScaleType	grScaleType (enumerated)	Specifies the type of scale used for the axis. Values are: Linear! Log10! Loge!
ScaleValue	grScaleValue (enumerated)	Specifies the scale of values on the axis. Values are: Actual! Cumulative! Percentage! CumPercent!
SecondaryLine	LineStyle (enumerated)	Specifies the type of secondary line for the axis. See DropLines in this table for the list of values.
ShadeBackEdge	Boolean	Specifies whether the back edge of the axis is shaded. Values are: TRUE – Shaded. FALSE – Not shaded.

Functions

grAxis function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

grDispAttr object

The PowerBuilder class grDispAttr is used to specify the appearance of text objects on a graph. There are grDispAttr objects for graph Titles, Legends, Pie Graph text, and two (DispAttr and LabelDispAttr) for each of the three axes (Category, Series, and Value) in a graph.

To refer to a property of a grDispAttr property, use this syntax:

graphcontrolname.axisname.grdispattrname.property

For example:

```
gr_1.Series.DispAttr.Italic
gr_1.Category.LabelDispAttr.DisplayExpression
```

A grDispAttr object has no events.

Properties

grDispAttr property	Datatype	Description
Alignment	Alignment (enumerated)	Specifies the alignment of the text. Values are: Center! Justify! Left! Right!
Alignment	Boolean	Specifies whether the text element should be autosized according to the amount of text being displayed. Values are: TRUE – Autosize. FALSE – Do not autosize.
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisplayExpression	String	An expression whose value is the label for the graph component. The default expression is the property containing the text for the graph component. The expression can include the text property and add other variable text.
Escapement	Integer	Specifies the rotation for the baseline of the text in tenths of a degree. 0 is horizontal. A value of 900 rotates the text 90 degrees; a value of 2700 rotates the text 270 degrees.

grDispAttr property	Datatype	Description
FaceName	String	Specifies a typeface name (for example, arial or courier) to use for the text.
FillPattern	FillPattern (enumerated)	Specifies the fill pattern for the text. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower right.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set to be used. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) for the text. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (character spacing) for the text. Values are: Default! Fixed! Variable!
Format	String	Specifies the display format for the text.
Italic	Boolean	Specifies whether the text is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
TextColor	Long	Specifies the color of the text. The color is a numeric value: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the point size of the text. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.

grDispAttr property	Datatype	Description
Weight	Integer	Specifies the stroke weight of the text. Sample values are 400 for normal or 700 for bold.

Functions

grDispAttr function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the user object.

GroupBox control

A GroupBox is a box used to group related controls. For example, you can use a GroupBox to group a series of RadioButtons or CommandButtons. The user cannot select the GroupBox but can select controls within the GroupBox. If the GroupBox contains RadioButtons, the user can select only one RadioButton in the GroupBox at a time.

When you hide or show a GroupBox, PowerBuilder does not automatically hide or show the controls in the GroupBox.

Properties

GroupBox property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.

GroupBox property	Datatype	Description
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleLowered! StyleRaised!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the text of the GroupBox is visible or grayed. Values are: TRUE – Text is visible. FALSE – Text is grayed.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.

GroupBox property	Datatype	Description
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Indicates whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control title.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Indicates whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.

GroupBox property	Datatype	Description
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

GroupBox event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
Other	When a Windows message occurs that is not a PowerBuilder event

Functions

GroupBox function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SetFocus	Integer	Sets focus to the specified control.

GroupBox function	Datatype returned	Description
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers the specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

HProgressBar control

You can use a progress bar to indicate the progress of a lengthy operation, such as an installation program that copies a large number of files. The HProgressBar control is a horizontal rectangle that fills with the system highlight color as the operation progresses.

Properties

HProgressBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

HProgressBar property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
MaxPosition	Unsigned Integer	Specifies the value of the Position property when the progress bar is at the right edge of the control. This value can be different from the end of the control's range, set with the SetRange function.
MinPosition	Unsigned Integer	Specifies the value of the Position property when the progress bar is at the left edge of the control. This value can be different from the start of the control's range, set with the SetRange function.
Pointer	String	Specifies the name of the stock pointer or file containing the pointer used for the control.
Position	Integer	Specifies the value of the current position within the range of the control (set with the SetRange function). The control uses the range and the current position to determine the percentage of the progress bar to fill with the highlight color.
SetStep	Integer	Specifies a step increment for the progress bar. The default is 10.
SmoothScroll	Boolean	Specifies that the control displays as a smooth scrolling bar instead of the default segmented bar.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units).

HProgressBar property	Datatype	Description
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

HProgressBar event	Occurs
Clicked	When the left mouse button is pressed on the control
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the left mouse button is double-clicked on the control
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

HProgressBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
OffsetPos	Integer	Moves the control's current position by the amount specified.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.

HProgressBar function	Datatype returned	Description
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SetFocus	Integer	Sets the focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRange	Integer	Sets the range of the control. The control uses the range and the current position to determine the percentage of the progress bar to fill with the highlight color.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
StepIt	Integer	Moves the control's current position by the amount specified by the value of the SetStep property.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

HScrollBar control

An HScrollBar is a horizontal bar with arrows at either end and a scroll box. Typically, you use an HScrollBar control as a slider control employed by users to specify a value on a continuous scale, or as a way to display graphically information to the user.

Usage note
The HScrollBar control is not the horizontal scroll bar that displays to allow the user to scroll through information in a control.

Properties

HScrollBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
MaxPosition	Integer	Specifies the value of the Position property when the scroll box is at the right edge of the control.
MinPosition	Integer	Specifies the value of the Position property when the scroll box is at the left edge of the control.
Pointer	String	Specifies the name of the stock pointer or file containing the pointer used for the control.
Position	Integer	Specifies a value between MinPosition and MaxPosition specifying the position of the scroll box.
StdHeight	Boolean	Specifies whether PowerBuilder uses the standard height for the control. Values are: TRUE – Standard height used for control. FALSE – Standard height not enforced for the control.

HScrollBar property	Datatype	Description
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

HScrollBar event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LineLeft	When the left arrow of the control is clicked
LineRight	When the right arrow of the control is clicked
LoseFocus	When the control loses focus (becomes inactive)
Moved	When the scroll box is moved (use the Position property to determine the new location)
Other	When a Windows message occurs that is not a PowerBuilder event
PageLeft	When the open space to the left of the scroll box is clicked
PageRight	When the open space to the right of the scroll box is clicked
RButtonDown	When the right mouse button is pressed on the control

Functions

HScrollBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets the focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

HTrackBar control

Like a scroll bar, a trackbar is used as a scrolling control, but clicking on the trackbar slider moves it in discrete increments instead of continuously. The HTrackBar control has a series of tick marks along the bottom of the trackbar channel.

To enable this control to be used properly from the keyboard, you must add code to the LineLeft, LineRight, PageLeft, and PageRight events. The code you add should change the slider Position property by the appropriate value and then pass the new slider position to the object or objects you associate with the trackbar control. You must code the Moved event if you want the trackbar control to pass on the slider position after the slider is dragged with a mouse.

Usage note
Use a trackbar when you want the user to select a discrete value. For example, you can use a trackbar to enable a user to select a timer interval or the size of a window.

Properties

HTrackBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.

HTrackBar property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
LineSize	Integer	Specifies how far the slider moves in response to keyboard input from the arrow keys. Setting LineSize to 1 indicates moving 1 increment in the range of values specified by the MaxPosition and MinPosition properties.
MaxPosition	Integer	Specifies the value of the Position property when the slider is at the right edge of the control.
MinPosition	Integer	Specifies the value of the Position property when the slider is at the left edge of the control.
PageSize	Integer	Specifies how far the slider moves in response to clicking in the slider channel area or pressing the Page keys. The default size is the difference between the MaxPosition and MinPosition properties divided by 5.
Pointer	String	Specifies the name of the stock pointer or file containing the pointer used for the control.
Position	Integer	Specifies a value between MinPosition and MaxPosition specifying the position of the slider.
Slider	Boolean	Specifies whether or not the trackbar contains a slider.
SliderSize	Integer	Specifies the size of the slider on the trackbar. A value of 0 makes the slider the default size.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
TickFrequency	Integer	Specifies tick mark frequency. Setting TickFrequency to 1 indicates 1 tick mark for each increment in the trackbar range of values.
TickMarks	HTickMarks (enumerated)	Specifies where tickmarks should be displayed. Values are: HTicksOnBottom! HTicksOnTop! HTicksOnBoth! HTicksOnNeither!
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.

HTrackBar property	Datatype	Description
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

HTrackBar event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LineLeft	When the left arrow key is clicked
LineRight	When the right arrow key is clicked
LoseFocus	When the control loses focus (becomes inactive)
Moved	When the slider is moved (use the Position property to determine the new location)
Other	When a Windows message occurs that is not a PowerBuilder event
PageLeft	When the Page Up key is clicked or when mouse clicks are made to the left of the slider in the trackbar channel
PageRight	When the Page Down key is clicked or when mouse clicks are made to the right of the slider in the trackbar channel
RButtonDown	When the right mouse button is pressed on the control

Functions

HTrackBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.

HTrackBar function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SelectionRange	Integer	Sets a selection range for the trackbar. When you select a range, a blue line is drawn in the channel of the trackbar and two arrows are drawn where the tickmarks are placed to indicate the beginning and end of the selection range.
SetFocus	Integer	Sets the focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

Inet object

The Inet object provides the ability to display a Web page in the default browser, access the HTML for a specified page, and send data to a CGI, ISAPI, or NSAPI program.

Properties

Inet property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Events

Inet event	Occurs
Constructor	Immediately before the Open event occurs in the window.
Destructor	Immediately after the Close event occurs in the window.

Functions

Inet function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetURL	Integer	Returns HTML for the specified URL.
HyperLinkToURL	Integer	Opens the default Web browser, displaying the specified URL.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
PostURL	Integer	Performs an HTTP Post, allowing a PowerBuilder application to send a request through CGI, NSAPI, or ISAPI.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

InkEdit control

An InkEdit control collects pen input (ink) on a Tablet PC and converts it to text. It is used with the handwriting recognition engine (“recognizer”) that is part of the Tablet PC platform.

Using with animation features
InkEdit controls may not paint correctly when you use animation features.

Properties

InkEdit property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the alignment of text in the control. Values are: Center! Justify! Left! Right!
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the border style of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisplayOnly	Boolean	Specifies whether the text is display only and cannot be changed by the user. Values are: TRUE – Text cannot be changed by user. FALSE – Text can be changed by user.

InkEdit property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be clicked. FALSE – Control cannot be clicked.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Courier).
Factoid	String	Specifies a context for ink recognition. Set this property if the input data is of a known type, such as a date or Web address, to constrain the search for a recognition result.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet enumerated datatype on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the font pitch used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.

InkEdit property	Datatype	Description
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays. Values are: TRUE – Horizontal scroll bar displays. FALSE – Horizontal scroll bar does not display.
IgnorePressure	Boolean	A drawing attribute that specifies whether the drawn ink gets wider as the pressure of the pen tip on the tablet surface increases. Values are: TRUE – Pressure from the pen tip is ignored. FALSE – The width of the ink increases with the pressure of the pen tip (default).
InkAntiAliased	Boolean	A drawing attribute that specifies whether the foreground and background colors along the edge of the drawn ink are blended (antialiased) to make the stroke smoother and sharper. Values are: TRUE – The ink stroke appears smoother and sharper (default). FALSE – The ink stroke is not antialiased.
InkColor	Long	A drawing attribute that specifies the current ink color. The default color is black.
InkHeight	Integer	A drawing attribute that specifies the height of the side of the rectangular pen tip in pixels. The default is 1pixel.
InkMode	InkMode (enumerated)	Specifies whether ink collection is enabled and whether ink only or ink and gestures are collected. Values are: CollectInk! – Only ink is collected. CollectInkAndGestures! – Ink and gestures are collected (default). InkDisabled! – Ink collection is disabled (the control behaves like a multiline edit control).
InkTransparency	Integer	A drawing attribute that specifies the transparency of drawn ink. The range of values is from 0 for totally opaque (the default) to 255 for totally transparent.
InkWidth	Integer	A drawing attribute that specifies the width of the pen in pixels. The default is 53 pixels. If the IgnorePressure property is not set, the actual width varies between .5 times the value of the Width property for minimum pressure and 1.5 times its value for maximum pressure.
InsertAsText	Boolean	Specifies whether the ink is inserted as text or as ink. Values are: TRUE – The ink is inserted as text (default). FALSE – The ink is inserted as ink.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic (default).
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) that can be entered in the control (0 means unlimited).

InkEdit property	Datatype	Description
Modified	Boolean	Specifies whether the text in the control has been modified since it was opened or last saved. Modified is the control's "dirty" flag, indicating that the control is in an unsaved state. Values are: TRUE – The control has been modified. FALSE – The control has not been modified. When the first change is made to a newly opened or saved control, PowerBuilder sets the Modified property to true and triggers the Modified event.
PenTip	InkPenTip (enumerated)	A drawing attribute that specifies whether the pen tip is round or rectangular. Values are: PenTipBall! – The pen tip is round (default). PenTipRectangle! – The pen tip is rectangular.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RecognitionTimer	Long	Specifies the time period in milliseconds between the last ink stroke and the start of text recognition. The default is 2000 (two seconds).
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
Status	InkEditStatus (enumerated)	A read-only property available at runtime that provides the current status of the control so that the user does not need to monitor the Stroke event. Values are CollectingInk!, RecognizingInk!, and Idle!.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined (default).

InkEdit property	Datatype	Description
UseMouseForInput	Boolean	Specifies whether the mouse can be used for input on a Tablet PC. Values are: TRUE – The mouse can be used for input. FALSE – The mouse cannot be used for input (default).
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible (default). FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed on the right of the control. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal (default) or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

InkEdit event	Occurs
Constructor	When the object is created, immediately before the Open event occurs in the window.
Destructor	When the object is destroyed, immediately after the Close event occurs in the window.
DragDrop	When a dragged control is dropped on the control.
DragEnter	When a dragged control enters the control.
DragLeave	When a dragged control leaves the control.
DragWithin	When a dragged control is within the control.
Gesture	When a gesture has occurred.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
LoseFocus	When the control loses focus (becomes inactive).
Moved	When a control has been changed and loses focus.
Other	When a Windows message occurs that is not a PowerBuilder event.
RButtonDown	When the right mouse button is pressed on the control.

InkEdit event	Occurs
RecognitionResult	When a recognition has occurred.
Stroke	When a stroke has occurred.

Functions

InkEdit function	Datatype returned	Description
ClassName	String	Returns the name of the control.
Clear	Integer	Clears the selected text (if any) from the control (but does not place it in the clipboard).
Copy	Integer	Copies (but does not delete) the selected text (if any) from the control to the clipboard.
Cut	Integer	Cuts (deletes) the selected text (if any) from the control to the clipboard.
Drag	Integer	Starts or ends the dragging of an InkEdit item.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Hides the specified InkEdit item.
Move	Integer	Moves a control or object to a specified location.
Paste	Integer	Inserts the contents of the clipboard (if any) at the insertion point in the control.
PointerX	Integer	Determines the distance from the left edge of an object to the pointer location.
PointerY	Integer	Determines the distance from the top edge of an object to the pointer location.
Position	Integer	Returns the position of the insertion point in the control.
PostEvent	Boolean	Adds the event to the end of the event queue of an object.
Print	Integer	Includes this object in a print job. Only the part visible on the screen is printed.
Resize	Integer	Resizes a control to the specified dimensions.
RecognizeText	Integer	Specifies that ink recognition should occur.
ReplaceText	Integer	Replaces the currently selected text (if any) with the specified string. If no text is selected, the ReplaceText function inserts the text at the insertion point.
SelectedLength	Integer	Returns the length of the selected text (if any) in the control.
SelectedText	String	Returns a string with the selected text (if any) from the control.

InkEdit function	Datatype returned	Description
SelectText	Long	Selects the text specified by the starting position and length.
SetFocus	Integer	Sets focus for a specified object or control.
SetPosition	Integer	Sets the position of the InkEdit control in the front-to-back order within a window.
SetRedraw	Integer	Controls the automatic redraw of an object after its properties have changed.
Show	Integer	Makes an object or control visible if it is hidden. If the object is already visible, Show brings it to the top.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

InkPicture control

An InkPicture control collects pen input ("ink") on a Tablet PC and layers it on a picture. It is used to collect ink annotations to a picture or signatures. It has no handwriting recognition capability.

Using with animation features
InkPicture controls may not paint correctly when you use animation features.

Properties

InkPicture property	Datatype	Description
AutoErase	Boolean	Specifies whether the auto erase feature available on some styluses is turned on. Values are: TRUE – AutoErase is turned on. FALSE – AutoErase is turned off (default).
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.

InkPicture property	Datatype	Description
BorderStyle	BorderStyle (enumerated)	Specifies the border style of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CollectionMode	InkCollectionMode (enumerated)	Specifies whether ink only, gestures only, or ink and gestures are collected. Values are: GestureOnly! – Only gestures are collected. InkOnly! – Only ink is collected (default). InkAndGesture! – Ink and gestures are collected.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
DynamicRendering	Boolean	Specifies whether the ink is rendered (displayed in the control) as it is drawn. The default is true.
EditMode	InkPicEditMode	Specifies whether the editing mode of the control is set for drawing ink, editing ink, or deleting ink. Values are: InkPicDeleteMode! – Ink is deleted. InkPicInkMode! – Ink can be drawn (default). InkPicSelectMode! – Ink is selected for editing.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be clicked. FALSE – Control cannot be clicked.

InkPicture property	Datatype	Description
EraserMode	Integer	Specifies whether ink is removed by stroke or point. Values are: 0 – Ink is removed by stroke (default). 1 – Ink is removed by point.
EraserWidth	Integer	Specifies the width of the eraser pen tip in pixels. The default is 212 pixels.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HighContrastInk	Boolean	Specifies whether ink is rendered in a single color when the system is in high contrast mode and draws the selection rectangle and handles in high contrast. Values are: TRUE – Ink is rendered in a single color in high contrast mode (default). FALSE – Ink is not rendered in a single color in high contrast mode.
IgnorePressure	Boolean	A drawing attribute that specifies whether the drawn ink gets wider as the pressure of the pen tip on the tablet surface increases. Values are: TRUE – Pressure from the pen tip is ignored. FALSE – The width of the ink increases with the pressure of the pen tip (default).
InkAntiAliased	Boolean	A drawing attribute that specifies whether the foreground and background colors along the edge of the drawn ink are blended (antialiased) to make the stroke smoother and sharper. Values are: TRUE – The ink stroke appears smoother and sharper (default). FALSE – The ink stroke is not antialiased.
InkColor	Long	A drawing attribute that specifies the current ink color. The default color is black.
InkEnabled	Boolean	Specifies whether the InkPicture control collects pen input. Values are: TRUE – The control collects pen input (default). FALSE – The control does not collect pen input and no pen-related events fire.
InkFileName	String	Specifies the name of the file that the ink was loaded from.
InkHeight	Integer	A drawing attribute that specifies the height of the side of the rectangular pen tip in pixels. The default is 1.
InkTransparency	Integer	A drawing attribute that specifies the transparency of drawn ink. The range of values is from 0 for totally opaque (the default) to 255 for totally transparent.

InkPicture property	Datatype	Description
InkWidth	Integer	A drawing attribute that specifies the width of the pen in pixels. The default is 53. If the IgnorePressure property is not set, the actual width varies between .5 times the value of the Width property for minimum pressure and 1.5 times its value for maximum pressure.
MarginX	Integer	Specifies the x-axis margin around the control in PowerBuilder units. The default value is 0.
MarginY	Integer	Specifies the y-axis margin around the control in PowerBuilder units. The default value is 0.
PenTip	InkPenTip (enumerated)	A drawing attribute that specifies whether the pen tip is round or rectangular. Values are: PenTipBall! – The pen tip is round (default). PenTipRectangle! – The pen tip is rectangular.
PictureFileName	String	Specifies the name of a file that contains the picture for the control. The default is an empty string. Supported formats are BMP, GIF, JPEG, PNG, and TIF.
PictureSizeMode	DisplaySizeMode	Specifies how the picture is displayed in the control. Values are: InkPicAutoSize! – The control is resized to display the entire picture. InkPicCentered! – The picture is centered in the control. InkPicNormal! – The picture is displayed in the upper-left corner of the control and any part of the picture that does not fit in the control is clipped (default). InkPicStretched! – The picture is stretched to fill the control.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTipText	String	Specifies a PowerTip for the control.
Status	InkPicStatus	Read-only property available at runtime that provides the current status of the control so that the user does not need to monitor the Stroke event. Values are CollectingInk! and Idle!.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control.
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible (default). FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

InkPicture event	Occurs
Clicked	When the left mouse button is clicked in the control
Constructor	When the object is created, immediately before the Open event occurs in the window
Destructor	When the object is destroyed, immediately after the Close event occurs in the window
DoubleClicked	When the object is destroyed, immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
Gesture	When a gesture has occurred
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control
SelectionChanged	When an item in the control is selected
SelectionChanging	When the selection is changing. Return 1 to prevent the selection from changing or 0 to allow it
SizeChanged	When the control has been resized
Stroke	When a stroke has occurred

Functions

InkPicture function	Datatype returned	Description
ClassName	String	Returns the name of the control.
Drag	Integer	Starts or ends the dragging of a InkEdit item.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Hides the specified InkEdit item.
LoadInk	Integer	Loads ink from a file or blob.
LoadPicture	Integer	Loads a picture from a file or blob.
Move	Integer	Moves a control or object to a specified location.

InkPicture function	Datatype returned	Description
PointerX	Integer	Determines the distance from the left edge of an object to the pointer location.
PointerY	Integer	Determines the distance from the top edge of an object to the pointer location.
PostEvent	Boolean	Adds the event to the end of the event queue of an object.
Print	Integer	Includes this object in a print job. Only the part visible on the screen is printed.
ResetInk	Integer	Clears ink from the control.
ResetPicture	Integer	Clears a picture from the control.
Resize	Integer	Resizes a control to the specified dimensions.
Save	Integer	Saves a picture and optionally any overlay ink to a file or blob in one of the following formats: bitmap, JPEG, GIF, TIF, or PNG.
SaveInk	Integer	Saves ink to a file or blob in one of the following formats: Ink Serialized Format (ISF), GIF, Base-64 encoded ISF, or Base-64 encoded GIF.
SetFocus	Integer	Sets focus for a specified object or control.
SetPosition	Integer	Sets the position of the InkEdit control in the front-to-back order within a window.
SetRedraw	Integer	Controls the automatic redraw of an object after its properties have changed.
Show	Integer	Makes an object or control visible if it is hidden. If the object is already visible, Show brings it to the top.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

InternetResult object

The InternetResult object acts as a buffer, receiving and caching asynchronous data, as it is returned using the Internet in response to the GetURL and PostURL function calls. The InternetResult object also provides the ability to process this data.

To use an InternetResult object, create a standard class user object that defines an InternetData function to process the passed HTML.

Properties

InternetResult property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Events

InternetResult event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window

Functions

InternetResult function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
InternetData	Integer	Processes the HTML data returned by a GetURL or PostURL function
InternetStatus	Integer	Not used
PostEvent	Boolean	Adds an event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

Line control

A line drawing object is a single straight solid or dashed line.

Properties

Line property	Datatype	Description
BeginX	Integer	Specifies the X position of one end of the line (the distance from the left edge of the window), in PowerBuilder units.
BeginY	Integer	Specifies the Y position of one end of the line (the distance from the top of the window), in PowerBuilder units.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
EndX	Integer	Specifies the X position of the other end of the line (the distance from the left edge of the window), in PowerBuilder units.
EndY	Integer	Specifies the Y position of the other end of the line (the distance from the top of the window), in PowerBuilder units.
LineColor	Long	Specifies the numeric value of the line color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
LineStyle	LineStyle (enumerated)	Specifies the style of the line. Values are: Continuous! Dash! DashDot! DashDotDot! Dot! Transparent!
LineThickness	Integer	Specifies the thickness of the line, in PowerBuilder units. If LineThickness is greater than one pixel (about three PowerBuilder units), the LineStyle is Continuous!.
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.

Events

Error event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.

Functions

Line function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
Resize	Integer	Changes the length of the control (changes the settings of the BeginX BeginY, EndX, and EndY properties)
Show	Integer	Makes the control visible
TypeOf	Object	Returns the type of the control

ListBox control

A ListBox displays available options or values. If more options or values exist than can display in the ListBox at one time or the text exceeds the width of the ListBox, the ListBox has one or two (vertical or horizontal) scroll bars.

Properties

ListBox property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.

ListBox property	Datatype	Description
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisableNoScroll	Boolean	Specifies behavior of a scroll bar. Values are: TRUE – The scroll bar is always visible but is disabled when all the items can be accessed without it. FALSE – The scroll bar is displayed only if it is necessary (based on the number of items and the height of the list box).
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.

ListBox property	Datatype	Description
ExtendedSelect	Boolean	Specifies whether users can select multiple items in the list box at one time. Values are: TRUE – Users can select multiple items by clicking on an item and dragging the mouse up or down to select items; using Click or Shift+ Click to select a sequential group of items; or using Ctrl+ Click on multiple items. FALSE – Users cannot select multiple items. **Used with MultiSelect** The MultiSelect property allows users to select multiple items in a list box by simply clicking on the items. If MultiSelect = true and ExtendedSelect = true, then the behavior of ExtendedSelect takes precedence.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays. Values are: TRUE – Horizontal scroll bar displays. FALSE – Horizontal scroll bar does not display.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Item[ ]	String	Specifies the items in the control.

ListBox property	Datatype	Description
MultiSelect	Boolean	Specifies whether users can select multiple items in the ListBox at one time. Values are: TRUE – Users can select multiple items. FALSE – Users cannot select multiple items. **Used with ExtendedSelect** The MultiSelect property allows users to select multiple items in a list box by simply clicking on the items. If MultiSelect = true and ExtendedSelect = true, then the behavior of ExtendedSelect takes precedence.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
Sorted	Boolean	Specifies whether the items in the ListBox are sorted. Values are: TRUE – Items are sorted. FALSE – Items are not sorted.
TabOrder	Integer	Specifies the tab value of the control (0 means the user cannot tab to the control).
TabStop[ ]	Integer array	Specifies the positions of the tab stops in the ListBox. The tab stops are in character positions, and the tab stop delimiter is a space. If you assign a value to only the first tab stop, TabStop[1], the tab stops are equally spaced using the number of character positions specified for the first tab stop. If more than one tab stop is entered, tab stops are located in the positions specified. You can define 16 tab stops in the control; the default array is TabStop[8], with a tab stop every eight character positions.
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.

ListBox property	Datatype	Description
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed on the right of the ListBox. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

ListBox event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Controls message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control
SelectionChanged	When an item in the control is selected

Functions

ListBox function	Datatype returned	Description
AddItem	Integer	Adds a new item to the end of the ListBox. If the Sorted property of the control is true, the items are sorted again after the item is added. The AddItem function does not update the Item[ ] property of this control.
ClassName	String	Returns the name assigned to the control.
DeleteItem	Integer	Deletes the item indicated by the index from the ListBox.
DirList	Boolean	Populates the ListBox with a list of the files of the specified type that match the specified file pattern.
DirSelect	Boolean	Returns the current selection for the control and puts it in the specified variable.
Drag	Integer	Starts or ends the dragging of a control.
FindItem	Integer	Finds the first item in the ListBox (after the specified index) that begins with the specified string.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
InsertItem	Integer	Adds a new item to the ListBox before the item indicated by the index. If the Sorted property of the control is true, the items are sorted again after the item is added.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Reset	Integer	Removes all items from the control.
Resize	Integer	Changes the size of the control.
SelectedIndex	Integer	Returns the index of the item in the ListBox that is currently selected. If more than one item is selected, returns the index of the first selected item.
SelectedItem	String	Returns the text of the first selected item.

ListBox function	Datatype returned	Description
SelectItem	Integer	Finds and highlights an item in the control. Use Syntax 1 when you know the text of the item but not its position. Use Syntax 2 when you know the position of the item in the control's list or you want to clear the current selection. SelectItem has no effect on a ListBox whose MultiSelect property is true. Instead, use SetState to select items without affecting the selected state of other items in the list.
SetFocus	Integer	Sets focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
SetState	Integer	Sets the state (highlighted or not highlighted) of the item indicated by the specified index. SetState works only for multiselect controls (those for which the MultiSelect property is true).
SetTop	Integer	Scrolls the items in the control so that the item indicated by the specified index is at the top of the control.
Show	Integer	Makes the control visible.
State	Integer	Returns 1 if the item specified by the specified index is selected (highlighted) and 0 if the item is not selected.
Text	String	Returns the text of the item in the control identified by the specified index.
Top	Integer	Returns the index number of the item currently at the top of the control.
TotalItems	Integer	Returns the total number of items in the control.
TotalSelected	Integer	Returns the total number of items selected in the control.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

ListView control

A ListView displays list information to the user. Each item of the ListView consists of text and pictures, which can be manipulated during application runtime.

Properties

ListView property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the accelerator key assigned for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AutoArrange	Boolean	Specifies whether PowerBuilder arranges icons automatically in large and small icon views.
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the border style of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order.
ButtonHeader	Boolean	Specifies whether the column titles in report view appear as buttons.
CheckBoxes	Boolean	Specifies whether the state images are replaced by check boxes. The check boxes are set to unchecked by default. The ListView control processes mouse and keyboard input to toggle the checked state. Values are: • TRUE – Check boxes are displayed. • FALSE – Check boxes are not displayed. The state of an item's check box can be determined by checking the state picture index for the item: Unchecked = 1 Checked = 2
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

ListView property	Datatype	Description
DeleteItems	Boolean	Specifies whether the user can delete a ListView item from a ListView control by pressing Delete.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the ICO file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
EditLabels	Boolean	Specifies whether the user can edit the labels in a control by clicking on a selected item. Note that the item must be selected first, by clicking on it.
Enabled	Boolean	Specifies whether the control is enabled (can be clicked). Values are: TRUE – Control can be clicked. FALSE – Control cannot be clicked.
ExtendedSelect	Boolean	Specifies whether users can select multiple items in the list box at one time. Values are: TRUE – Users can select multiple items by selecting outside all items and dragging to create a rectangle enclosing the desired items; by using Click or Shift+ Click to select a sequential group of items; or by using Ctrl+ Click on multiple items. FALSE – Users cannot select multiple items.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Courier).
FixedLocations	Boolean	Specifies whether the user cannot drag items to new positions in a control.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.

ListView property	Datatype	Description
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the font pitch used for the text in the control. Values are: Default! Fixed! Variable!
FullRowSelect	Boolean	Specifies whether an entire row can be selected in report view. Values are: • TRUE – In report view, an entire row can be selected. • FALSE – In report view, just the item in the first column can be selected.
GridLines	Boolean	Specifies whether the report view displays gridlines: • TRUE – In report view, gridlines are displayed. • FALSE – In report view, gridlines are not displayed.
HeaderDragDrop	Boolean	Specifies whether column headers can be dragged to move columns in report view: • TRUE – In report view, column headers can be dragged. • FALSE – In report view, dragging column headers does not move the columns.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HideSelection	Boolean	Specifies whether selected text stays selected (highlighted) even when the control does not have focus. Values are: TRUE – Text does not stay highlighted. FALSE – Text stays highlighted.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Item[ ]	String	Specifies the items in the control. Not updated after initialization.
ItemPictureIndex[ ]	Integer	Identifies the picture associated with the item. The picture index itself is associated with a specific icon, bitmap, or cursor. Not updated after initialization.

ListView property	Datatype	Description
LabelWrap	Boolean	Specifies whether long labels wrap under the ListView item in a large icon view. Values are: TRUE – Labels wrap. FALSE – Labels do not wrap. LabelWrap does not apply to list, report, or small icon views.
LargePictureHeight	Integer	Specifies the size, in pixels, for the height of the picture used in the large icon view. In a script, this value can be set only before a large picture has been added to the large picture index list. If the large picture height is 0, PowerBuilder uses the height of the first picture added to the large picture index.
LargePictureMaskColor	Long	Specifies the color to be transparent when used in a large icon view. This color is used when the picture is added at initialization or with the function AddLargePicture.
LargePictureName[ ]	String	Specifies the name of the picture used in large icon view. The picture can be an icon, cursor, or bitmap supplied by the user or a stock picture from the PowerBuilder library. Not updated after initialization.
LargePictureWidth	Integer	Specifies the size, in pixels, for the width of the picture used in the large icon view. In a script, this value can be set only before a large picture has been added to the large picture index list. If the large picture width is 0, PowerBuilder uses the width of the first picture added to the large picture index.
LayoutRTL	Boolean	Specifies that the layout of the control should be a mirror image of the standard layout. Values are: TRUE – Elements in the control are right justified. FALSE – Elements in the control are left justified (default).
OneClickActivate	Boolean	Specifies whether one click initiates the ItemActivate event: • TRUE – One click fires the ItemActivate event, causes the item to change color as the mouse moves over it (hot tracking), and causes the mouse to change to a hand cursor when it is over the item. • FALSE – The item does not turn color as the mouse moves over it (assuming that TrackSelect = false) and the mouse does not change to a hand cursor when it is over the item (assuming that TwoClickActivate = false). However, the ItemActivate event is always initiated when an item is double-clicked, even though OneClickActivate = false and TwoClickActivate = false.

ListView property	Datatype	Description
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order (default).
Scrolling	Boolean	Specifies whether the user can scroll vertically when not all of the items in a control are visible. Values are: TRUE – Scrolling is enabled. FALSE – Scrolling is disabled.
ShowHeader	Boolean	Specifies whether column titles appear in a report view. Values are: TRUE – Titles appear in a report view. FALSE – Titles do not appear in a report view.
SmallPictureHeight	Integer	Specifies the size, in pixels, for the height of the picture used in the small icon view. In a script, this value can be set only before a small picture has been added to the small picture index list. If the small picture height is 0, PowerBuilder uses the height of the first picture added to the small picture index.
SmallPictureMaskColor	Long	Specifies the color to be transparent when used in a small icon view. Used when the picture is added at initialization or with the function AddSmallPicture.
SmallPictureName[ ]	String	Specifies the name of the picture used in small icon view. The picture can be an icon, cursor, or bitmap supplied by the user or a stock picture from the PowerBuilder library. Not updated after initialization.
SmallPictureWidth	Integer	Specifies the size, in pixels, for the width of the picture used in the small icon view. In a script, this value can be set only before a small picture has been added to the small picture index list. If the small picture width is 0, PowerBuilder uses the width of the first picture added to the small picture index.
SortType	grSortType	Specifies whether items are sorted alphabetically based on the item label. Values are: Ascending! Descending! UserDefinedSort! Unsorted!

ListView property	Datatype	Description
StatePictureHeight	Integer	Specifies the size, in pixels, for the height of the state picture. In a script, this value can be set only before a state picture has been added to the state picture index list. If the state picture height is 0, PowerBuilder uses the height of the first picture added to the state picture index list.
StatePictureMaskColor	Long	Specifies the color to be transparent when used in a state picture. Used when the picture is added at initialization or with the function AddStatePicture.
StatePictureName[ ]	String	Specifies the name of the picture used as the state picture. The picture can be an icon, cursor, or bitmap supplied by the user or a stock picture from the PowerBuilder library. Not updated after initialization.
StatePictureWidth	Integer	Specifies the size, in pixels, for the width of the state picture. In a script, this value can be set only before a state picture has been added to the state picture index list. If the state picture width is 0, PowerBuilder uses the width of the first picture added to the state picture index list.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
TrackSelect	Boolean	Specifies whether items appear in a different color when the mouse moves over them (hot tracking) and whether an item is selected if the mouse pauses over it. Values are: TRUE – An item changes color when the mouse moves over it, and an item is selected if the mouse pauses over it. FALSE – An item does not change color nor is it selected when the mouse moves over or pauses on it.

ListView property	Datatype	Description
TwoClickActivate	Boolean	Specifies whether two clicks initiate the ItemActivate event: TRUE – Clicking twice (one click to select the item, one click to activate) fires the ItemActivate event, causes the item to change color as the mouse moves over it (hot tracking), and causes the mouse to change to a hand cursor when it is over the item. FALSE – The item does not turn color as the mouse moves over it (assuming that TrackSelect = false) and the mouse does not change to a hand cursor when it is over the item (assuming that OneClickActivate = false). However, the ItemActivate event is always initiated when an item is double-clicked, even though TwoClickActivate = false.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
UnderlineCold	Boolean	When the OneClickActivate property is true, this property specifies whether hot tracking (color of items changes when mouse moves over them) is turned on and items not highlighted are underlined. TRUE – Hot tracking is turned on and nonhighlighted items are underlined. FALSE – Nonhighlighted items are not underlined.
UnderlineHot	Boolean	When either the OneClickActivate or TwoClickActivate property is true, this property specifies whether hot tracking (color of items changes when mouse moves over them) is turned on and items that are highlighted are underlined. TRUE – Hot tracking is turned on and highlighted items are underlined. FALSE – Highlighted items are not underlined.
View	ListViewView	Specifies the layout of the ListBox. Valid values are: ListViewLargeIcon! – Items are arranged from left to right. Uses large pictures. ListViewSmallIcon! – Items are arranged from left to right. Uses small pictures. ListViewList! – Items are arranged from top to bottom. Uses small pictures. ListViewReport! – Items are arranged from top to bottom. Uses small pictures. Additional columns of information can be associated with each item. At least one column must be created to view data in this view.

ListView property	Datatype	Description
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

ListView event	Occurs
BeginDrag	When the user begins a drag operation with the left mouse button. If the DragAuto property is set to true, the drag begins automatically. If the DragAuto property is set to false, the drag operation must be done programmatically.
BeginLabelEdit	When the user starts to edit a ListView item label. Return 1 to prevent setting to the new text. Return 0 to accept the new text.
BeginRightDrag	When the user begins a drag operation with the right mouse button. If the DragAuto property is set to true, the drag begins automatically. If the DragAuto property is set to false, the drag operation must be done programmatically.
Clicked	When the control is clicked.
ColumnClick	When the column is clicked
Constructor	When the object is created, immediately before the Open event occurs in the window.
DeleteAllItems	When all items in a ListView are deleted.
DeleteItem	When a ListView item is deleted.
Destructor	When the object is destroyed, immediately after the Close event occurs in the window.
DoubleClicked	When the control is double-clicked.
DragDrop	When a dragged control is dropped on the ListView control.
DragEnter	When a dragged control enters the control, including entering the narrow border around the display area.
DragLeave	When a dragged control leaves the control, including leaving by crossing into the tab page display area.
DragWithin	When a dragged control is within the control but not on a ListView item.
EndLabelEdit	When the user finishes editing a ListView item label. Return 1 to prevent setting to the new text. Return 0 to accept the new text.

ListView event	Occurs
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
InsertItem	When an item is being inserted.
ItemActivate	When an item is double-clicked, or when the item is single-clicked if the property OneClickActivate = true, or when the item is clicked twice if the property TwoClickActivate = true.
ItemChanged	When an item has changed.
ItemChanging	When an item is changing. Return 1 to prevent the change, or 0 to accept the change.
Key	When the user presses a key.
LoseFocus	When the control loses focus (becomes inactive).
Other	When a Windows message occurs that is not a PowerBuilder event.
RightClicked	When the control is right-clicked.
RightDoubleClicked	When the control is right-double-clicked.
Sort	When two items are compared. Return codes: 1 – if item 1 > item 2. 0 – if item 1 = item 2. -1 – if item 1 < item 2.

Functions

ListView function	Datatype returned	Description
AddColumn	Integer	Adds a column to a ListView control report view.
AddItem	Integer	Adds an item to a ListView control.
AddLargePicture	Integer	Adds an icon, cursor, or bitmap to the large image list.
AddSmallPicture	Integer	Adds an icon, cursor, or bitmap to the small image list.
AddStatePicture	Integer	Adds an icon, cursor, or bitmap to the state image list.
Arrange	Integer	Arranges the items in a ListView control large or small icon view.
ClassName	String	Returns the name of the control.
DeleteColumn	Integer	Deletes a column from a ListView control.
DeleteColumns	Integer	Deletes all columns from a ListView control.
DeleteItem	Integer	Deletes an item from a ListView control.
DeleteItems	Integer	Deletes all items from a ListView control.
DeleteLargePicture	Integer	Deletes a specified icon, cursor, or bitmap from the large image list.

ListView function	Datatype returned	Description
DeleteLargePictures	Integer	Deletes all icons, cursors, and bitmaps from the large image list.
DeleteSmallPicture	Integer	Deletes a specified icon, cursor, or bitmap from the small image list.
DeleteSmallPictures	Integer	Deletes all icons, cursors, and bitmaps from the small image list.
DeleteStatePicture	Integer	Deletes a specified icon, cursor, or bitmap from the state image list.
DeleteStatePictures	Integer	Deletes all icons, cursors, and bitmaps from the large state list.
Drag	Integer	Starts or ends the dragging of a ListView item.
EditLabel	Integer	Starts editing a specific ListView item label.
FindItem	Integer	Searches for the next item that satisfies the specified search criteria.
GetColumn	Integer	Syntax 1: Does not apply to a ListView control. Syntax 2: Returns the properties of a specified column in a ListView control report view.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetItem	Integer	Retrieves information for a specified item.
GetOrigin	Integer	Finds the X and Y coordinates of the upper-left corner of the ListView item.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Hides the specified ListView item.
InsertColumn	Integer	Inserts a column into a ListView control report view.
InsertItem	Integer	Inserts an item into a ListView control.
Move	Integer	Moves a control or object to a specified location.
PointerX	Integer	Determines the distance from the left edge of an object to the pointer location.
PointerY	Integer	Determines the distance from the top edge of an object to the pointer location.
PostEvent	Boolean	Adds the event to the end of the event queue of an object.
Print	Integer	Includes this object in a print job. Only the part visible on the screen is printed.
Resize	Integer	Resizes a control to the specified dimensions.
SelectedIndex	Integer	Returns the number of the selected item in a ListView control.
SetColumn	Integer	Syntax 1: Does not apply to a ListView control. Syntax 2: Sets the properties of a particular column in a ListView control report view.
SetFocus	Integer	Sets focus for a specified object or control.
SetItem	Integer	Sets the values for a given ListView item.
SetOverlayPicture	Integer	Maps a picture index to an overlay picture index. Only four overlay picture indexes are available.

ListView function	Datatype returned	Description
SetPosition	Integer	Sets the position of the ListView control in the front-to-back order within a window.
SetRedraw	Integer	Controls the automatic redraw of an object after its properties have changed.
Show	Integer	Makes an object or control visible if it is hidden. If the object is already visible, Show brings it to the top.
Sort	Integer	Sorts the items in a ListView control.
TotalColumns	Integer	Returns the number of columns in a ListView control report view.
TotalItems	Integer	Returns the number of items in a ListView control.
TotalSelected	Integer	Returns the number of selected items in a ListView control.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

ListViewItem object

A ListViewItem object is a system structure that populates a ListView control. ListViewItems have no events.

Properties

ListViewItem property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CutHighlighted	Boolean	Specifies whether the item is the target of a cut operation. Values are: TRUE – The item is the target of a cut operation. FALSE – The item is not the target of a cut operation.
Data	Any	Assigns any user-defined data to a ListView item.
DropHighlighted	Boolean	Specifies whether the item is the target of a DragDrop operation. Values are: TRUE – The item is the target of a DragDrop operation. FALSE – The item is not the target of a DragDrop operation.

ListViewItem property	Datatype	Description
HasFocus	Boolean	Specifies whether the item has focus. Values are: TRUE – The item has focus. FALSE – The item does not have focus.
ItemX	Integer	Identifies the X location of the item relative to the upper-left corner of the control.
ItemY	Integer	Identifies the Y location of the item relative to the upper-left corner of the control.
Label	String	Identifies the string label associated with the item.
OverlayPictureIndex	Integer	Identifies the overlay picture associated with the item.
PictureIndex	Integer	Identifies the large and small picture associated with the item.
Selected	Boolean	Specifies whether the item is selected. Values are: TRUE – The item is selected. FALSE – The item is not selected.
StatePictureIndex	Integer	Identifies the state picture associated with the item.

Functions

ListViewItem function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the parent of the object.
TypeOf	Object	Returns the type of the object.

mailFileDescription object

The mailFileDescription object is a system structure containing information about an attachment file to a mail message. A mailFileDescription object has no events.

Properties

mailFileDescription property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
FileType	mailFileType (enumerated)	Specifies the type of the attachment file. Values are: mailAttach! – data file. mailOLE! – embedded OLE object. mailOLEStatic! – static OLE object. Only mailAttach! is supported for Extended MAPI. If you specify the other value (mailOLE! or mailOLEStatic!), it will work as mailAttach!
Filename	String	Specifies the name of the attachment file.
Pathname	String	Specifies the full path of the attachment file including the file name.
Position	Unsignedlong	Specifies the position of the attachment file within the message body. Required when sending multiple attachments.

Functions

mailFileDescription function	Datatype returned	Description
ClassName	String	Returns the class of an object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

mailMessage object

The mailMessage object is a system structure containing information about a specific mail message. A mailMessage object has no events.

Properties

mailMessage property	Datatype	Description
AttachmentFile[ ]	mailFileDescription	Specifies the file attachment for the current message. A mailFileDescription array contains information about an attachment file.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ConversationID	String	Specifies the conversation thread ID for the current message.
DateReceived	String	Indicates the date on which the current message was received.
MessageSent	Boolean	Indicates whether the current message has been sent to the mail server. (Read-only at runtime only.) TRUE – Message has been sent to mail server. FALSE – Message has not yet been sent.
MessageType	String	Indicates the type of the current message. A value other than null or an empty string indicates use by an application other than interpersonal mail. (Runtime only.)
NoteText	String	Specifies the content of the message body. (Runtime only.)
ReceiptRequested	Boolean	Indicates whether a return receipt is requested for the current message. (Runtime only.) TRUE – Return receipt requested. FALSE – Return receipt not requested.
Recipient[ ]	mailRecipient	Specifies the recipient of the current message. For mailSend, mailOriginator! is not a valid value for the Recipient property. The valid values are mailto!, mailcc!, and mailbcc!. To specify that the sender receive a copy of the message, use mailcc!.
Subject	String	Specifies the subject line, displayed in the message header, for the current message.
Unread	Boolean	Indicates whether or not the message has been read. (Read-only at runtime only.) TRUE – Message has not been read. FALSE – Message has been read.

Functions

mailMessage function	Datatype returned	Description
ClassName	String	Returns the class of the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

mailRecipient object

The mailRecipient object is a system structure containing information about the recipient of a mail message. You populate the mailRecipient structure using the mailAddress function. A mailRecipient object has no events.

For information about the mailAddress function, see the *PowerScript Reference*.

Properties

mailRecipient property	Datatype	Description
Address	String	Specifies the electronic mail address of the current mail recipient. (Runtime only.)
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
EntryID	Blob	Binary entry identifier information used internally.
Name	String	Specifies the name of the current mail recipient. (Runtime only.)
RecipientType	mailRecipientType (enumerated)	Specifies the type of the current mail recipient. Values are: mailBCC! mailCC! mailOriginator! mailTo!

Functions

mailRecipient function	Datatype returned	Description
ClassName	String	Returns the class of the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

mailSession object

The mailSession nonvisual object signs on and establishes a messaging application program interface (MAPI) session.

Properties

mailSession property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
MessageID[ ]	String	Contains the IDs of the messages in a user's mail inbox
SessionID	Long	Contains the handle of the current messaging session

Events

mailSession event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

mailSession function	Datatype returned	Description
ClassName	String	Returns the class of the object

mailSession function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
mailAddress	mailReturnCode	Updates the mailRecipient array for a mail message
mailDeleteMessage	mailReturnCode	Deletes a mail message from the user's electronic mail inbox
mailGetMessages	mailReturnCode	Populates the MessageID array of a mailSession object with the message IDs in the user's inbox
mailHandle	UnsignedLong	Obtains the handle of a mailSession object
mailLogoff	mailReturnCode	Ends the mail session, breaking the connection between the PowerBuilder application and mail
mailLogon	mailReturnCode	Establishes a mail session for the PowerBuilder application
mailReadMessage	mailReturnCode	Opens a mail message whose ID is stored in the mail session's message array
mailRecipientDetails	mailReturnCode	Displays a dialog box with the specified recipient's address information
mailResolveRecipient	mailReturnCode	Obtains a valid electronic mail address based on a partial or full user name and optionally updates information in the system's address list if the user has privileges to do so
mailSaveMessage	mailReturnCode	Creates a new message in the user's inbox or replaces an existing message
mailSend	mailReturnCode	Sends a mail message
PostEvent	Integer	Adds an event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specified event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

MDIClient object

An MDI window is a frame window in which you can open multiple document windows (sheets) and move among the sheets.

An MDIClient object is the area in which open sheets display in a standard MDI frame. In a standard MDI window, PowerBuilder sizes the MDIClient so that it fills the space inside the frame. For example, if the frame has a menu bar and MicroHelp, the MDIClient fills the space between the sides of the frame and the space below the menu bar and above the MicroHelp.

In a custom MDI window, you determine the size of the client area. For example, when a frame has buttons below the menu bar in the frame, you size the client area so it begins below the buttons.

An MDIClient object has no associated events.

Properties

MDIClient property	Datatype	Description
BackColor	Long	Specifies the numerical value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BringToTop	Boolean	Specifies whether PowerBuilder moves the MDIClient object to the top of the front-to-back order. Values are: TRUE – Object moved to top. FALSE – Object not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Height	Integer	Specifies the height of the MDIClient object, in PowerBuilder units.
MicroHelpHeight	Integer	Holds the height of the MicroHelp in the MDIClient object. If the style of the MDI window is MDI Frame (no MicroHelp), MicroHelpHeight is 0; otherwise, it is the height of the MicroHelp. You cannot set the value of this property.
Tag	String	Specifies the tag value assigned to the MDIClient object.
Visible	Boolean	Specifies whether the MDIClient object is visible. Values are: TRUE – Object is visible. FALSE – Object is not visible.
Width	Integer	Specifies the width of the MDIClient object, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the MDI frame window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the MDI frame window), in PowerBuilder units.

Functions

MDIClient function	Datatype returned	Description
ClassName	String	Returns the name assigned to the MDIClient object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the MDIClient object invisible.
Move	Integer	Moves the MDIClient object to a specified location.
Resize	Integer	Changes the size of the MDIClient object.
SetRedraw	Integer	Controls automatic redrawing of the MDIClient object after each change in its properties. To reduce flicker when the user closes all the sheets, set SetRedraw to FALSE.
Show	Integer	Makes the MDIClient object visible.
TypeOf	Object	Returns the type of the object.

Menu object

In PowerBuilder, you use the Menu painter to create menus and toolbars. Typically, menus are lists of items (usually commands or options) that a user can select in the currently active window. Menus can display in a menu bar, in a drop-down or cascading menu, or as pop-up menus. A toolbar is associated with a menu, and its toolbar buttons act as shortcuts for choosing items from the menu.

PowerBuilder provides a Menu system object used to develop menus. A Menu object can contain other Menu objects that appear, for example, as the items in a drop-down or cascading menu. When the user clicks a Menu object, a Clicked event is triggered. If there is a drop-down or cascading menu under the clicked object, the script for the Clicked event for the object is executed, and then the menu displays. If there is no menu under the object, the script for the Clicked event for the object is executed.

Menus that you import or migrate from earlier versions of PowerBuilder use the Traditional menu style by default. Menus with a Contemporary style have a three-dimensional menu appearance similar to those in Microsoft Office 2003 and Visual Studio 2005, and can include bitmap and menu title bands.

Properties

Menu property	Datatype	Description
BitmapBackColor	Long	Background color of the bitmap band of the menu. (Default is silver.)
BitmapGradient	Boolean	Background of the bitmap band to a gradient style.
Checked	Boolean	Specifies whether the Menu object is selected. The state of the checked property is reflected in the toolbar button. Values are: TRUE – Object is selected. FALSE – Object is not selected.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Default	Boolean	Specifies whether the menu item is the default and appears in a bold typeface. Only one item within a menu should be set to default. In context menus, the Default property is used to indicate the action that would have been performed if the user had double-clicked on the object rather than right-clicked on it. The property is also used to indicate which operation would have been performed if the item had been dragged with the left mouse button rather than the right mouse button. Values are: TRUE – Menu item is bolded. (Valid on Windows XP and Windows 2003.) FALSE – Menu item is not bolded.
Enabled	Boolean	Specifies whether the Menu object is enabled (can be clicked). The state of the enabled property is reflected in the toolbar button. Values are: TRUE – Object is enabled. FALSE – Object is not enabled.
FaceName	String	Font face name.
Italic	Boolean	Italic font.
Item[ ]	Menu	Specifies the Menu objects under a Menu object.
MenuAnimation	Boolean	Visual sizing cue to the menu item bitmap when the associated menu item is selected. This property is ignored if the MenuImage property is not assigned.
MenuBackColor	Long	Background color of the menu.
MenuBitmaps	Boolean	Bitmap band for the menu.
MenuHighlightColor	Long	Menu highlight color. The default is the default Windows highlight color.

Menu property	Datatype	Description
MenuImage	String	Bitmap image to be used with the menu item. This property is ignored if the MenuBitmaps property for the menu object is not selected or is set to false. If you change MenuImage at runtime, the height of the image does not change, therefore you should avoid assigning a larger or smaller bitmap dynamically.
MenuItemType	MenuItemType (enumerated)	Allows you to identify special Menu objects that are used differently on different platforms. Values are: MenuItemTypeAbout! MenuItemTypeExit! MenuItemTypeHelp! MenuItemTypeNormal!
MenuStyle	MenuStyle (enumerated)	Overall menu style. Values are: contemporarymenu! and traditionalmenu!
MenuTextColor	Long	Menu text color. (Default is the Windows menu text color.)
MenuTitles	Boolean	Menu title band.
MenuTitleText	String	Label for a menu item that has a cascading submenu. The label text is set vertically in a column to the left of the submenu items and the bitmaps for submenu items, if any. If the vertical label text is longer than the height of all the submenu items, the label text is cut from the end. This property is ignored if the MenuTitles property for the menu object is not selected.
MergeOption	MenuMergeOption (enumerated)	Specifies how the object is affected when an OLE object is activated. Values are: EditMenu! Exclude! FileMenu! HelpMenu! Merge! WindowMenu! For more information about MergeOption, see the chapter on using OLE in *Application Techniques*.
MicroHelp	String	Specifies the text of the MicroHelp for the object.
ParentWindow	Window	Specifies the window that owns the Menu object.
ShiftToRight	Boolean	Specifies whether the Menu object shifts down or to the right when other Menu objects are added in a descendent menu. Values are: TRUE – Object shifts to right. FALSE – Object shifts down.

Menu property	Datatype	Description
Shortcut	Integer	Specifies the shortcut key for the Menu object. This property should be set only in the Menu painter. It cannot be set at runtime.
Tag	String	Specifies the tag value assigned to the Menu object.
Text	String	Specifies the text in the Menu object.
TextSize	Integer	Font character size in points for menu items. This property does not apply to the Traditional menu style, and it does not apply to the main menu bar, which has a fixed height of 8 points.
TitleBackColor	Long	Background color of the title panel.
TitleGradient	Boolean	Background gradient style for the title panel.
ToolbarAnimation	Boolean	Specifies animation for the toolbar image. You can select the ToolbarAnimation check box on the Toolbar tab in the Properties view for each menu item unless you are using the traditional toolbar style for the current menu object. If you do not select an image for the ToolbarItemName property of a menu item, the selection you make for the ToolbarAnimation property is ignored.
ToolbarBackColor	Long	Background color of the menu toolbar.
ToolbarGradient	Boolean	Gradient of the menu toolbar background.
ToolbarHighlightColor	Long	Highlight color for the toolbar buttons when they are selected.
ToolbarItemBarIndex	Integer	Specifies which toolbar the Menu object is on when multiple toolbars exist. If setting this index results in the object being the first item on a new toolbar, the toolbar is implicitly created. If setting this index results in emptying a toolbar, the toolbar is implicitly destroyed.
ToolbarItemDown	Boolean	Specifies how the toolbar button appears. Values are: TRUE – Toolbar button appears down. FALSE – Toolbar button appears up. This property is automatically reset when any button is pressed using the mouse.
ToolbarItemDownName	String	Specifies the name of the toolbar icon associated with the Menu object when it is down.
ToolbarItemName	String	Specifies the name of a stock toolbar picture you want to use to represent an item in the toolbar or a string containing the name of a bitmap file.
ToolbarItemOrder	Integer	Specifies the order of the item in the toolbar.
ToolbarItemSpace	Integer	Specifies the amount of empty space before the item in the toolbar.

Menu property	Datatype	Description
ToolbarItemText	String	Specifies the text that displays in the toolbar item when the display text option is on for toolbars.
ToolbarItemVisible	Boolean	Specifies whether the toolbar item displays. Values are: TRUE – The toolbar item is visible. FALSE – The toolbar item is not visible. If any toolbar item has a ToolbarItemName assigned, an empty toolbar displays even if this property has been set to false for all toolbar items. To control display of the toolbar, use the ToolbarVisible property of the window.
ToolbarStyle	ToolbarStyle (enumerated)	Overall style of the menu toolbar. Values are: contemporarytoolbar! and traditionaltoolbar!
ToolbarTextColor	Long	Color of the text in menu toolbar.
Underline	Boolean	Underline font.
Visible	Boolean	Specifies whether the Menu object is visible. Values are: TRUE – Object is visible. FALSE – Object is not visible.
Weight	Integer	Font weight.

Events

Menu event	Occurs
Clicked	When the Menu object is clicked (selected or unselected)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
Selected	When the user moves to the Menu object using the arrow keys or the mouse

Functions

Menu function	Datatype returned	Description
Check	Integer	Displays a check mark next to the Menu object and sets the Checked property.
ClassName	String	Returns the class of the Menu object.
Disable	Integer	Disables (and grays) the Menu object so that it cannot be selected and unsets the Enabled property.

Menu function	Datatype returned	Description
Enable	Integer	Enables the Menu object so that it can be selected and displays it normally (not grayed) and sets the Enabled property.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the Menu object invisible.
PopMenu	Integer	Displays the Menu object at the specified location.
PostEvent	Integer	Adds an event to the end of the message queue for the Menu object.
Show	Integer	Makes the Menu object visible.
TriggerEvent	Integer	Triggers a specified event in the Menu object and executes the script for the event.
TypeOf	Object	Returns the type of the control.
Uncheck	Integer	Removes the check mark next to the Menu object and sets the Checked property to FALSE.

MenuCascade object

Menu objects contained within a MenuCascade object appear as a drop-down button palette.

Properties

MenuCascade property	Datatype	Description
Checked	Boolean	Specifies whether the Menu object is selected. The state of the checked property is reflected in the toolbar button. Values are: TRUE – Object is selected. FALSE – Object is not selected.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Columns	Integer	Specifies the number of columns in the drop-down button palette.
CurrentItem	Menu	Specifies the Menu object that is currently displayed on the main toolbar as the representative (active) button for the drop-down button palette.

MenuCascade property	Datatype	Description
Default	Boolean	Specifies whether the menu item is the default and appears in a bold typeface. Only one item within a menu should be set to default. In context menus, the Default property is used to indicate the action that would have been performed if the user had double-clicked on the object rather than right-clicked on it. The property is also used to indicate which operation would have been performed if the item had been dragged with the left mouse button rather than the right mouse button. Values are: TRUE – Menu item is bolded. FALSE – Menu item is not bolded.
DropDown	Boolean	Specifies whether the Menu objects contained in the MenuCascade object display as a drop-down button palette. Values are: TRUE – Menu objects contained in the MenuCascade object are displayed as a drop-down button palette. FALSE – Menu objects contained in the MenuCascade object are displayed as normal toolbar items.
Enabled	Boolean	Specifies whether the Menu object is enabled (can be clicked). The state of the enabled property is reflected in the toolbar button. Values are: TRUE – Object is enabled. FALSE – Object is not enabled.
Item[ ]	Menu	Specifies the Menu objects under a Menu object.
MenuItemType	MenuItemType (enumerated)	Allows you to identify special menu items that are used differently on different platforms. Values are: MenuItemTypeAbout! MenuItemTypeExit! MenuItemTypeHelp! MenuItemTypeNormal!
MergeOption	MenuMerge-Option (enumerated)	Specifies how the object is affected when an OLE 2.0 object is activated. Values are: EditMenu! Exclude! FileMenu! HelpMenu! Merge! WindowMenu! For more information about MergeOption, see the chapter on using OLE in *Application Techniques*.

MenuCascade property	Datatype	Description
MicroHelp	String	Specifies the text of the MicroHelp for the object.
ParentWindow	Window	Specifies the window that owns the Menu object.
ShiftToRight	Boolean	Specifies whether the Menu object shifts down or to the right when other Menu objects are added in a descendent menu. Values are: TRUE – Object shifts to right. FALSE – Object shifts down.
Shortcut	Integer	Specifies the shortcut key for the object. This property should be set only in the Menu painter. It cannot be set at runtime.
Tag	String	Specifies the tag value assigned to the Menu object.
Text	String	Specifies the text in the Menu object.
ToolbarItemDown	Boolean	Specifies how the toolbar button appears. Values are: TRUE – Toolbar button appears down. FALSE – Toolbar button appears up. This property is automatically reset when any button is pressed using the mouse.
ToolbarItemDown Name	String	Specifies the name of the toolbar icon associated with the Menu object when it is down.
ToolbarItemBarIndex	Integer	Specifies which toolbar the object is on when multiple toolbars exist. If setting this index results in the item being the first item on a new toolbar, the toolbar is implicitly created. If setting this index results in emptying a toolbar, the toolbar is implicitly destroyed.
ToolbarItemName	String	Specifies the name of a stock toolbar picture you want to use to represent an item in the toolbar or a string containing the name of a bitmap file.
ToolbarItemOrder	Integer	Specifies the order of the item in the toolbar.
ToolbarItemSpace	Integer	Specifies the amount of empty space before the item in the toolbar.
ToolbarItemText	String	Specifies the text that displays in the toolbar item when the display text option is on for toolbars.
ToolbarItemVisible	Boolean	Specifies whether the toolbar item displays. Values are: TRUE – Toolbar item is visible. FALSE – Toolbar item is not visible.
Visible	Boolean	Specifies whether the Menu object is visible. Values are: TRUE – Object is visible. FALSE – Object is not visible.

Events

MenuCascade event	Occurs
Clicked	When the Menu object is clicked (selected or unselected)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
Selected	When the user moves to the Menu object using the arrow keys or the mouse

Functions

MenuCascade function	Datatype returned	Description
Check	Integer	Displays a check mark next to the Menu object and sets the Checked property
ClassName	String	Returns the class of the Menu object
Disable	Integer	Disables (and grays) the Menu object so that it cannot be selected and unsets the Enabled property
Enable	Integer	Enables the Menu object so that it can be selected, displays it normally (not grayed), and sets the Enabled property
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	Power Object	Returns a reference to the name of the parent object
Hide	Integer	Makes the Menu object invisible
PopMenu	Integer	Displays the Menu object at the specified location
PostEvent	Integer	Adds an event to the end of the message queue for the Menu object
Show	Integer	Makes the Menu object visible
TriggerEvent	Integer	Triggers a specified event in the Menu object and executes the script for the event
TypeOf	Object	Returns the type of the control
Uncheck	Integer	Removes the check mark next to the Menu object and sets the Checked property to FALSE

Message object

The Message object is used to process events that are not defined by PowerBuilder, to communicate parameters between windows when you open and close them, and to specify whether optional parameters are used in TriggerEvent or PostEvent.

You can also customize your own version of the Message object by defining a class user object inherited from the built-in Message object.

For more information about creating a custom Message object, see the chapter on user objects in the *PowerBuilder Users Guide.*

Properties

The first four properties of the Message object correspond to the first four properties of the Microsoft Windows message structure:

Message property	Datatype	Description
Handle	Long	The handle of the window or control.
Number	UnsignedInt	The number that identifies the event (this number comes from Windows).
WordParm	Long	The word parameter for the event (this parameter comes from Windows). The parameter's value and meaning are determined by the event.
LongParm	Long	The long parameter for the event (this number comes from Windows). The parameter's value and meaning are determined by the event.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DoubleParm	Double	A numeric or a numeric variable.
StringParm	String	A string or a string variable.
PowerObjectParm	PowerObject	Any PowerBuilder object type including structures.

Message property	Datatype	Description
Processed	Boolean	A boolean value set in the script for the user-defined event or the Other event. Values are: TRUE – The script processed the event; do not call the default window process (DefWindowProc) after the event has been processed. FALSE – (Default) Call DefWindowProc after the event has been processed.
ReturnValue	Long	When Message.Processed is true, specifies the value you want returned to Windows. This property is ignored when Message.Processed is false.

Events

Message event	Occurs
Constructor	When the user object is created
Destructor	When the user object is destroyed

Functions

Message function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue of the user object
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event
TypeOf	Object	Returns the type of the user object

MLSync object

An MLSync object is derived from the MLSynchronization object base class. Its primary function is to launch *dbmlsync.exe*, a separate process that synchronizes a SQL Anywhere remote database with a consolidated database. The MLSync properties control the options used for the synchronization. MLSync events represent callbacks that are automatically triggered by *dbmlsync.exe* at specific times during the synchronization process.

Properties

MLSync property	Datatype	Description
AdditionalOpts	String	Used to pass additonal dbmlsync options.
AuthenticateParms	String	Used with authenticate_parameters connection script. Equivalent to the `-ap "parm1,parm2"` dbmlsync option.
DataSource	String	The ODBC data source name used to connect to a SQL Anywhere remote database. Equivalent to the `-c "dsn=myDSN"` dbmlsync option, where *myDSN* is the data source name.
DBPass	String	Password for the SQL Anywhere remote database. Equivalent to the `-c "pwd=myPassword"` dbmlsync option, where *myPassword* is the password for the database connection.
DBUser	String	User ID for the SQL Anywhere remote database. Equivalent to the `-c "uid=myUserID"` dbmlsync option, where *myUserID* is the user ID for the database connection.
EncryptionKey	String	Encryption key for SQL Anywhere remote database. Equivalent to the `-c "dbkey=myKey"` dbmlsync option, where *myKey* is the encryption key for the database.
ErrorText	String	Contains error and diagnostic messages generated when an MLSync function is called incorrectly.
ExtendedOpts	String	Extended dbmlsync options. Equivalent to the `-e "extops"` dbmlsync option.
Host	String	The machine name for the synchronization server. Equivalent to the `-e "adr='host=machineName'"` dbmlsync option, where *machineName* is the host machine.
LogFileName	String	Creates a log file at this path if UseLogFile=true. Equivalent to the `-o logfilename` dbmlsync option, where *logfilename* is the full name of the log file you want to create.
LogOpts	String	Additional dbmlsync options to control logging output.

MLSync property	Datatype	Description
MLPass	String	The MobiLink™ password needed to connect to the synchronization server. Equivalent to the `-mp password` dbmlsync option.
MLServerVersion	Long	The version of the SQL Anywhere remote database and synchronization server.
MLUser	String	The MobiLink user name needed to connect to the synchronization server. Equivalent to the `-u username` dbmlsync option.
ObjectRevision	Long	You increment this value each time you rebuild the MLSync object with new default property values. At runtime, if a new ObjectRevision value is detected, the default property values are written to the Windows registry and used to initialize MLSync. For subsequent synchronizations, MLSync is initialized using the property values obtained from the Windows registry.
Port	String	The port number for the synchronization server. Equivalent to the `-e "adr='port=portno'"` dbmlsync option, where *portno* is the number of the port you use for synchronization.
ProcessOption	SyncProcessType (enumerated)	Sets the direction for synchronization. Values are DownloadOnly!, UploadOnly!, and Bidirectional! (default).
ProgressWindowName	String	Specifies the class name of a progress window generated by the MobiLink Synchronization Wizard. The progress window is an optional selection in the synchronization wizard, or a customized user-defined window. It should contain the same callback events as the MLSync object. An MLSync object that is generated by the wizard automatically triggers the appropriate window event in order to display synchronization progress.
Publication	String	The publication to be processed. If more than one publication is specified, you must separate each name with a comma. Equivalent to the `-n pub1,pub2` dbmlsync option.
SyncRegistryKey	String	Location in the Windows registry where MLSync property values from a previous synchronization are stored. The MobiLink Synchronization Wizard generates a synchronization options window that allows an end user to customize the MLSync properties at runtime and save these settings when the SetSyncRegistryProperties function is triggered.
UseLogFile	Boolean	If true, creates a synchronization log.
UseWindow	Boolean	Used at runtime by the objects that are generated by the synchronization wizard. When set to true, the generated functions create a progress window to display status information about the synchronization process.
WindowObject	Window	Instance of a synchronization progress window. The class name of WindowObject must match the ProgressWindowName value.

Events

MLSync event	Occurs
BeginDownload	At the beginning of the download procedure
BeginLogScan	Before dbmlsync scans the transaction log to assemble the upload
BeginSync	At the beginning of the synchronization
BeginUpload	At the beginning of the upload procedure
ConnectMobiLink	When dbmlsync connects to the MobiLink server
DisconnectMobiLink	Immediately after disconnecting from the synchronization server
DisplayMessage	On display of an informational message
EndDownload	At the end of download processing
EndLogScan	After the scan of the transaction log completes for upload
EndSync	At the end of synchronization
EndUpload	After transmission of the upload to the synchronization server
ErrorMessage	On display of an error messag
FileMessage	On display of a file message
ProgressIndex	Periodically whenever the synchronization triggers updates to a progress bar
SyncPreview	Returns generated dbmlsync command arguments immediately prior to launching the process
UploadAck	On completion of upload processing
WaitForUploadAck	When the synchronization process starts a new waiting period for upload acknowledgement
WarningMessage	On display of a warning message

Functions

MLSync function	Datatype returned	Description
CancelSync	Integer	Cancels a synchronization that is in progress
GetCommandString	String	Returns the command string that is generated from the current MLSync property values
GetDbmlsyncPath	String	Returns the full path and file name of *dbmlsync.exe* that is installed on the workstation.
GetObjectRevisionFromRegistry	Integer	Returns the current synchronization build number from the Windows registry
GetSyncRegistryProperties	Integer	Retrieves the ObjectRevision property saved in the Windows registry by a previous call to SetSyncRegistryProperties
SetNewMobiLinkPassword	Integer	Changes the MobiLink password on the consolidated database

MLSync function	Datatype returned	Description
SetParm	Integer	Passes SyncParm object properties to an MLSync object
SetSyncRegistryProperties	Integer	Writes synchronization property values to the Windows registry
Synchronize	Integer	Launches a synchronization process using the MLSync command string properties that have been set

MLSynchronization object

The MLSynchronization object is an abstract class from which MLSync objects are derived.

Properties

MLSynchronization property	Datatype	Description
AdditionalOpts	String	Used to pass additonal dbmlsync options.
AuthenticateParms	String	Used with authenticate_parameters connection script.
ErrorText	String	Contains error and diagnostic messages generated when a function is called incorrectly.
ExtendedOpts	String	Extended dbmlsync options.
Host	String	The machine name for the synchronization server.
LogFileName	String	Creates a log file at this path if UseLogFile=true.
LogOpts	String	Additional dbmlsync options to control logging output.
MLPass	String	The MobiLink password passed to the synchronization server.
MLServerVersion	Long	The version of the SQL Anywhere remote database and synchronization server.
MLUser	String	The MobiLink user name passed to the synchronization server.
ObjectRevision	Long	At runtime, if a new ObjectRevision value is detected, the default property values are written to the Windows Registry and used to initialize objects of type MLSync. For subsequent synchronizations, MLSync is initialized using the property values obtained from the Windows registry.
Port	String	The port number for the synchronization server.

MLSynchronization property	Datatype	Description
ProcessOption	SyncProcessType (enumerated)	Sets the direction for synchronization. Values are DownloadOnly!, UploadOnly!, and Bidirectional! (default).
ProgressWindowName	String	Name of a user-defined customized window or a window generated by the MobiLink synchronization wizard to indicate synchronization progress.
Publication	String	The publication to be processed.
SyncRegistryKey	String	Location in the Windows registry where MLSync property values from a previous synchronization are stored.
UseLogFile	Boolean	If true, creates a synchronization log.
UseWindow	Boolean	Used by the synchronization wizard to determine whether to create a progress window to display status information about the synchronization process.
WindowObject	Window	Synchronization progress window.

Events

MLSynchronization event	Occurs
BeginDownload	At the beginning of the download procedure
BeginSync	At the beginning of the synchronization
BeginUpload	At the beginning of the upload procedure
ConnectMobiLink	When the MobiLink synchronization server connects to the consolidated database server
DisconnectMobiLink	Immediately after disconnecting from the synchronization server
DisplayMessage	On display of an informational message
EndDownload	At the end of download processing
EndSync	At the end of synchronization
EndUpload	After transmission of the upload to the synchronization server
ErrorMessage	On display of an error message
FileMessage	On display of a file message
ProgressIndex	Periodically during synchronization after updates to a synchronization progress bar
SyncPreview	Returns generated dbmlsync command arguments immediately prior to launching the process
UploadAck	On completion of upload processing
WaitForUploadAck	When the synchronization process starts a new waiting period for upload acknowledgement

MLSynchronization event	Occurs
WarningMessage	On display of a warning message

Functions

MLSynchronization function	Datatype returned	Description
CancelSync	Integer	Cancels a synchronization that is in progress
GetObjectRevisionFromRegistry	Integer	Retrieves the ObjectRevision property saved in the Windows registry by a previous call to SetSyncRegistryProperties
GetSyncRegistryProperties	Integer	Retrieves synchronization property values from the Windows registry
SetNewMobiLinkPassword	Integer	Changes the MobiLink password on the consolidated database
SetParm	Integer	Passes SyncParm object properties to a synchronization object
SetSyncRegistryProperties	Integer	Writes synchronization property values to the Windows registry
Synchronize	Integer	Launches a synchronization process using the command string properties that have been set

MonthCalendar control

A MonthCalendar control provides a calendar-like user interface that makes it easy for users to enter or select dates.

Properties

MonthCalendar property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.

MonthCalendar property	Datatype	Description
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AutoSize	Boolean	Specifies whether the calendar is sized automatically to hold a single month. Values are: TRUE – The calendar sizes to hold a single month (default). FALSE – The calendar does not size to hold a single month.
BackColor	Long	Specifies the numeric value of the background color of the control: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to put the control into Drag mode manually by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.

MonthCalendar property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier). This property does not work on the Windows 7/8.1/10 operating system.
FirstDayOfWeek	WeekDay (enumerated)	Specifies which day of the week displays on the left in the calendar.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic. This property does not work on the Windows 7/8.1/10 operating system.
MaxSelectCount	Integer	Specifies the maximum number of days the user can select from the calendar. The default is 1.
MonthBackColor	Long	Specifies the numeric value of the background color of a month: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.

MonthCalendar property	Datatype	Description
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order.
ScrollRate	Integer	Specifies the number of months the calendar scrolls when the user clicks a scroll button.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control.
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the numeric value of the color used for text within a month: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10. This property does not work on the Windows 7/8.1/10 operating system.
TitleBackColor	Long	Specifies the numeric value of the background color of the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
TitleTextColor	Long	Specifies the numeric value of the color used for text in the calendar's title: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
TodayCircle	Boolean	Specifies whether the border of today's date on the calendar displays in red. Values are: TRUE – The Today circle is displayed (default). FALSE – The Today circle is not displayed.
TodaySection	Boolean	Specifies whether the label "Today:" followed by the current date displays at the bottom of the calendar. Values are: TRUE – The Today section is displayed (default). FALSE – The Today section is not displayed. This property does not work correctly on the Windows 7/8.1/10 operating system.

MonthCalendar property	Datatype	Description
TrailingTextColor	Long	Specifies the numeric value of the color used for leading and trailing days in the calendar: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property does not work on the Windows 7/8.1/10 operating system.
Underline	Boolean	Specifies whether text is underlined. This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
WeekNumbers	Boolean	Specifies whether a number representing the number of the week in the year displays to the left of each row in the calendar. Values are: TRUE – Week numbers are displayed. FALSE – Week numbers are not displayed (default).
Weight	Integer	This property is not relevant in MonthCalendar controls.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

MonthCalendar event	Occurs
Clicked	When the control is clicked (selected) with the left mouse button
Constructor	Immediately before the Open event occurs in the window
DateChanged	Immediately after a date is selected
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is clicked twice with the left mouse button
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control

MonthCalendar event	Occurs
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Controls message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

MonthCalendar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
ClearBoldDates	Integer	Clears bold format of dates in calendar
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetDateLimits	Integer	Retrieves the maximum and minimum date limits specified for the calendar
GetDisplayRange	Integer	Retrieves the date range of displayed months and returns the number of displayed months
GetParent	PowerObject	Returns a reference to the name of the parent object
GetSelectedDate	Integer	Retrieves the selected date
GetSelectedRange	Integer	Retrieves the range of selected dates
GetToday	Date	Returns the date that the calendar uses as today's date
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds the specified event to the end of the event queue for the specified object
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetBoldDate	Integer	Displays the specified date in bold
SetDateLimits	Integer	Sets the maximum and minimum date limits for the calendar
SetFocus	Integer	Sets focus to the specified control

MonthCalendar function	Datatype returned	Description
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
SetSelectedDate	Integer	Selects a specified date
SetSelectedRange	Integer	Sets the range of selected dates
SetToday	Integer	Sets the value that is used by the calendar as today's date
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

MultiLineEdit control

A MultiLineEdit control is a box in which the user can enter and edit more than one line of text. You typically use a MultiLineEdit as an input field.

Properties

MultiLineEdit property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the text alignment in the control. Values are: Center! Justify! Left! Right!

MultiLineEdit property	Datatype	Description
AutoHScroll	Boolean	Specifies whether the control automatically scrolls horizontally when data is entered or deleted. Values are: TRUE – Control automatically scrolls horizontally. FALSE – Control does not automatically scroll horizontally.
AutoVScroll	Boolean	Specifies whether the control automatically scrolls vertically when data is entered or deleted. Values are: TRUE – Control automatically scrolls vertically. FALSE – Control wraps.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisplayOnly	Boolean	Specifies whether the text is display only and cannot be changed by the user. Values are: TRUE – Text cannot be changed by user. FALSE – Text can be changed by user.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.

MultiLineEdit property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Tahoma).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HideSelection	Boolean	Specifies whether selected text stays selected (highlighted) even when the control does not have focus. Values are: TRUE – Text does not stay highlighted. FALSE – Text stays highlighted.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays. Values are: TRUE – Horizontal scroll bar displays. FALSE – Horizontal scroll bar does not display.

MultiLineEdit property	Datatype	Description
IgnoreDefaultButton	Boolean	Specifies whether the Clicked event for the window's Default command button is triggered when user presses Enter. Values are: TRUE – Do not trigger Clicked event; add new line in MultiLineEdit control. FALSE – (Default) Trigger Clicked event; do not add new line in MultiLineEdit control.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) that can be entered in the control (0 means unlimited).
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
TabStop[ ]	Integer	Specifies the positions of the tab stops in the control. The tab stops are in character positions, and the tab stop delimiter is a space. If you assign a value to only the first tab stop, TabStop[1], the tab stops are equally spaced using the number of character positions specified for the first tab stop. If more than one tab stop is entered, tab stops are located in the positions specified. You can define 16 tab stops in the control; the default array is TabStop[8], with a tab stop every eight character positions.
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextCase	TextCase (enumerated)	Specifies the case in which text entered in the control displays. Values are: AnyCase! Lower! Upper!
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.

MultiLineEdit property	Datatype	Description
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed on the right of the control. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

MultiLineEdit event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Modified	When a control has been changed and loses focus
Other	When a Controls message occurs that is not a PowerBuilder event

MultiLineEdit event	Occurs
RButtonDown	When the right mouse button is pressed on the control

Functions

MultiLineEdit function	Datatype returned	Description
CanUndo	Boolean	Returns true if the Undo function can be used to undo the last edit in the control and returns false if it cannot
ClassName	String	Returns the name assigned to the control
Clear	Integer	Clears the selected text (if any) from the control (but does not place it in the clipboard)
Copy	Integer	Copies (but does not delete) the selected text (if any) from the control to the clipboard
Cut	Integer	Cuts (deletes) the selected text (if any) from the control to the clipboard
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
LineCount	Integer	Returns the number of lines in the MultiLineEdit
LineLength	Integer	Returns the length of the line in which the insertion point is positioned
Move	Integer	Moves the control to a specified location
Paste	Integer	Inserts the contents of the clipboard (if any) at the insertion point in the control
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
Position	Integer	Returns the position of the insertion point in the control
PostEvent	Boolean	Adds the specified event to the end of the event queue for the specified object
Print	Integer	Prints the control
ReplaceText	Integer	Replaces the currently selected text (if any) with the specified string. If no text is selected, the ReplaceText function inserts the text at the insertion point
Resize	Integer	Changes the size of the control

MultiLineEdit function	Datatype returned	Description
Scroll	Integer	Moves the contents of the control up or down by the specified number of lines
SelectedLength	Integer	Returns the length of the selected text (if any) in the control
SelectedLine	Integer	Returns the number of the line in which the insertion point is currently located
SelectedStart	Integer	Returns the starting position of the selected text (if any) in the control
SelectedText	String	Returns a string with the selected text (if any) from the control
SelectText	Integer	Selects the text specified by the starting position and length
SetFocus	Integer	Sets focus to the specified control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TextLine	String	Returns the entire text of the line in which the insertion point is currently located
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control
Undo	Integer	Cancels the previous editing function performed in the control

OLEControl control

An OLEControl placed in a window can contain an object, such as a spreadsheet or word processing document, that was created by an OLE-aware application. The PowerBuilder application's user can activate the object and edit it in the application in which it was created (the server application).

OLE controls are displayed in the OLE tab of the Browser as Insertable Objects.

For more information about using OLE in an application, see *Application Techniques*.

Properties

OLEControl property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Activation	omActivation	Specifies how the OLE object will be activated.
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BinaryIndex	Integer	Internal use only.
BinaryKey	String	Internal use only.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.

OLEControl property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassLongName	String	(Read-only) The long name for the server application associated with the OLE object in the control.
ClassShortName	String	(Read-only) The short name for the server application associated with the OLE object in the control.
ContentsAllowed	omContentsAllowed	Specifies whether the OLE object in the control must be embedded or linked or whether either method is allowed when Insert is called at runtime.
DisplayName	String	User-readable name for your OLE control when the control is activated in place. This name is displayed in OLE dialog boxes and windows that show the object's name. If you do not specify a value, the name of the control (such as ole_1) is used for DisplayName.
DisplayType	omDisplayType	Specifies how the OLE object will be displayed in the control. The control can display the actual contents or an icon to represent the object, or the document can be displayed as an ActiveX document.
DocFileName	String	(Read-only) The name of the file containing the object. If the object has never been saved to a disk file, then the value of this property is " ".
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.

OLEControl property	Datatype	Description
FocusRectangle	Boolean	Specifies whether a dotted rectangle (the focus rectangle) frames the control when it has focus. Values are: TRUE – Control framed when it has focus. FALSE – Control not framed when it has focus.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
IsDragTarget	Boolean	Specifies whether an OLE object can be dropped on the control. Values are: TRUE – OLE object can be dropped on control. FALSE – OLE object cannot be dropped on control.
LinkItem	String	(Read-only) The entire link name of the item to which the object is linked. For example, if the object is linked to *C:\FILENAME.XLS!A1:B2*, then LinkItem would contain `C:\FILENAME.XLS!A1:B2`.
LinkUpdateOptions	omLinkUpdate Options	Specifies how a linked object in the control is updated. If automatic, the link is updated when the object is opened and whenever the object changes in the server application. If manual, the link is not updated.
Object	omObject	Used in scripts to apply server commands to the linked or embedded OLE object within the control.
ObjectData	Blob	If the object is embedded, the object itself is stored as a blob in the ObjectData property. If the object is linked, this property contains the link information and the cached image (for display).
ParentStorage	omStorage	(Read-only) Specifies the parent storage.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
Resizable	Boolean	Specifies whether the control is resizable. Values are: TRUE – Control is resizable. FALSE – Control is not resizable.
SizeMode	SizeMode (enumerated)	Specifies the size mode for the OLE document. Choices are: Clip! – The object's image displays full size. If it is larger than the OLE control, the image is clipped by the control's borders. Stretch! – The object's image is resized to fit into and fill the OLE container control. This is the default value.
TabOrder	Integer	Specifies tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.

OLEControl property	Datatype	Description
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

OLEControl event	Occurs
Clicked	When the control is clicked (selected or unselected)
Close	Just before a window is removed from display (closed)
Constructor	Immediately before the Open event occurs in the window
DataChange	When the server application notifies the control that data has changed
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (and possibly activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
Error	During OLE automation when an error occurs
ExternalException	During OLE automation when the OLE server generates an exception during command execution (getting and setting properties, calling functions)
GetFocus	Just before the control receives focus (and possibly becomes activated)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive).
Other	When an operating environment message occurs that is not a PowerBuilder event
PropertyChanged	When an OLE Server supporting notifications sends this message to the control
PropertyRequestEdit	When an OLE Server supporting notifications sends this message to the control
RButtonDown	When the right mouse button is pressed in the control
Rename	When the server application notifies the control that the object has been renamed
Save	When the server application notifies the control that the data has been saved

OLEControl event	Occurs
SaveObject	When the server application saves the object in the control
ViewChange	When the server application notifies the control that the view shown to the user has changed

Functions

OLEControl function	Datatype returned	Description
Activate	Integer	Activates the object server either in place or offsite
ClassName	String	Returns the name assigned to the control
Clear	Integer	Releases the OLE object and deletes references to it without updating storage
Copy	Integer	Copies the contents of the control to the clipboard
Cut	Integer	Copies the contents of the control to the clipboard and clears the control
DoVerb	Integer	Executes the specified verb
Drag	Integer	Puts the object into drag mode
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetData	Integer	Returns data in a caller-supplied format from an OLE server that supports Uniform Data Transfer
GetNativePointer	Integer	Returns a pointer to the underlying OLE object
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
InsertClass	Integer	Inserts a new object created from a class ID
InsertFile	Integer	Inserts a new object where the source is a template file
InsertObject	Integer	Presents the user with a standard dialog box and inserts based on the selection
LinkTo	Integer	Links to a file and (optionally) an item within the file
Move	Integer	Moves the control to a specified location
Open	Integer	Opens a document file and loads the object into the control or opens a substorage within the previously opened storage and loads an object
Paste	Integer	Pastes the contents of the clipboard into the control
PasteLink	Integer	Pastes a link to the contents of the clipboard into the control
PasteSpecial	Integer	Presents a dialog box allowing the user to select Paste or PasteLink

OLEControl function	Datatype returned	Description
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
ReleaseNativePointer	Integer	Releases the pointer to the underlying OLE object
Resize	Integer	Changes the size of the control
Save	Integer	Saves an object previously loaded from a storage
SaveAs	Integer	Saves the contained object as a member in the requested storage or saves the contained object to the requested storage file
SelectObject	Integer	Sets the internal state of the control (updates menu)
SetData	Integer	Sends data in a caller-supplied format to an OLE server that supports Uniform Data Transfer
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specific event for the control and executes the script for the event
TypeOf	Object	Returns the type of the control
UpdateLinksDialog	Integer	Invokes the OLE dialog to update link information if the file has moved

OLECustomControl control (OCX)

The PowerBuilder class OLECustomControl is a container for OLE custom controls, also known as ActiveX controls or OCXs. When you create a PowerBuilder OLE custom control container, the Insert Object dialog prompts you to select the control to insert in the container. Your choices are the controls that have been registered in the system registry. If a control is not registered by its install process, you can register it in the Insert Object dialog box.

The Browser also displays registered controls. Select the OLE tab of the Browser and double-click OLE Custom Controls.

Properties

In OLE terminology, font information and the display name are called **ambient properties**. Ambient properties provide default information that the custom control can use, if it is programmed to recognize and use such information.

PowerBuilder does not display text for the control, so it does not use the font and display name properties directly. If the control is programmed to recognize ambient properties, it can use the values PowerBuilder provides when the control displays text or needs a name to display in a title bar.

OLECustom Control property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the text alignment in the control. Values are: Center! Justify! Left! Right!
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BinaryIndex	Integer	Internal use only.
BinaryKey	String	Internal use only.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order (set at runtime only). Values are: TRUE – Control moved to top. FALSE – Control not moved to top.

OLECustom Control property	Datatype	Description
Cancel	Boolean	Specifies whether the control acts as the Cancel button in the window (the Cancel button receives a Clicked event if the user presses Esc). Values are: TRUE – Control is the Cancel button. FALSE – Control is not the Cancel button.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassLongName	String	Specifies the long name for the server application associated with the OLE object in the control (read-only).
ClassShortName	String	Specifies the short name for the server application associated with the OLE object in the control (read-only).
DisplayName	String	Specifies a user-readable name for your OLE control. This name is displayed in OLE dialog boxes and windows that show the object's name. If you do not specify a value, the name of the control (such as ole_1) is used for DisplayName.
Default	Boolean	Specifies whether the button-style OLE control is the default control in the window (the default control has a thick border and receives a Clicked event if the user presses Enter without selecting a control). This property applies only to controls that act like command buttons. Values are: TRUE – Control is the default control. FALSE – Control is not the default control. **Editable controls** Default behavior can be affected by editable controls on the window. For more information, see the *PowerBuilder Users Guide*.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.

OLECustom Control property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies a typeface name (for example, arial or courier) that you want the control to use for text (when the control is designed to use this ambient property).
FocusRectangle	Boolean	Specifies whether a dotted rectangle (the focus rectangle) frames the control when it has focus. Values are: TRUE – Control framed when it has focus. FALSE – Control not framed when it has focus.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set that you want the control to use for text (when the control is designed to use this ambient property). For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) that you want the control to use for text (when the control is designed to use this ambient property). Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (character spacing) that you want the control to use for text (when the control is designed to use this ambient property). Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.

OLECustom Control property	Datatype	Description
IsDragTarget	Boolean	Specifies whether data can be dropped on the control. Values are: TRUE – Data can be dropped on control. FALSE – Data cannot be dropped on control.
Italic	Boolean	Specifies that you want the control to display text in italic (when the control is designed to use this ambient property). Values are: TRUE – Text is italic. FALSE – Text is not italic.
Object	omObject	Specifies the link information that connects the control to the server's data.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
TabOrder	Integer	Specifies tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the color that you want the control to use for text (when the control is designed to use this ambient property). The color is a numeric value: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the point size that you want the control to use for displaying text (when the control is designed to use this ambient property). For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies that you want the control to underline text (when the control is designed to use this ambient property). Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight that you want the control to use for text (when the control is designed to use this ambient property). Sample values are 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

The PowerBuilder Script painter displays PowerBuilder events and events defined by the control inserted in the control.

To get information about an event that does not appear in this list, check the documentation for the control.

OLECustomControl event	Occurs
Clicked	When the control is clicked (selected or unselected).
Constructor	Immediately before the Open event occurs in the window.
DataChange	When the server application notifies the control that data has changed.
Destructor	Immediately after the Close event occurs in the window.
DoubleClicked	When the control is double-clicked (and possibly activated).
DragDrop	When a dragged control is dropped on the control.
DragEnter	When a dragged control enters the control.
DragLeave	When a dragged control leaves the control.
DragWithin	When a dragged control is within the control.
Error	During OLE automation when an error occurs.
ExternalException	During OLE automation when the OLE server generates an exception during command execution (getting and setting properties, calling functions).
GetFocus	Just before the control receives focus (and possibly becomes activated).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
LoseFocus	When the control loses focus (becomes inactive).
Other	When an operating environment message occurs that is not a PowerBuilder event.
PropertyChanged	When an OLE Server supporting notifications sends a message that a property value has been changed.
PropertyRequestEdit	When an OLE Server supporting notifications sends a message that a property value is about to be changed.
RButtonDown	When the right mouse button is pressed in the control.

Functions

OLECustomControl function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Puts the object into drag mode

OLECustomControl function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetData	Integer	Returns data in a format you specify from an OLE server that supports Uniform Data Transfer
GetNativePointer	Integer	Returns a pointer to the underlying OLE object
GetParent	Power Object	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
ReleaseNativePointer	Integer	Releases pointer to underlying OLE object
Resize	Integer	Changes the size of the control
SetData	Integer	Sends data in a caller-supplied format to an OLE server that supports Uniform Data Transfer
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specific event for the control and executes the script for the event
TypeOf	Object	Returns the type of the control

OLEObject object

The OLEObject object acts as a proxy for a remote OLE object.

You can customize your own version of the OLEObject object by defining a standard class user object inherited from the built-in OLEObject. You can then access the OLEObject Constructor, Destructor, Error, and ExternalException events by writing scripts that contain code for the events.

Coding Error and ExternalException events
If you code the Error and ExternalException events, any active exception handler for a RuntimeError will not be processed. However, you can throw an exception in the scripts for these events, and you can make the arguments of these events available for exception handling by putting the arguments in a string in a throw statement that passes the string to the exception handler.

OLEObject objects are displayed in the OLE tab of the Browser as Programmable Objects.

For more information about creating a custom OLEObject object, see the chapter on user objects in the *PowerBuilder Users Guide*.

For more information about using the OLEObject object in an application, see *Application Techniques*.

OLEObject is a dynamic object
In order to support OLE, OLEObject is a dynamic object. The PowerBuilder compiler acceps property names and function names and parameter lists that are not already defined for the object. If the properties or functions do not exist during execution, you get a runtime error.

Properties

OLEObject property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Handle	Objhandle	Internal use only.

Events

OLEObject event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.
Error	When an error is found in a data or property expression for an external object.
ExternalException	When the evaluation of an expression involving properties of an external object causes an error. This type of event occurs before the Error event.

Functions

OLEObject function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object
ConnectToNewObject	Integer	Creates a new instance of the class and connects to it
ConnectToNewRemoteObject	Integer	Creates a new OLE object in the specified remote server application and associates the new object with a PowerBuilder OLEObject variable
ConnectToObject	Integer	Opens a specified file and connects to the corresponding server application
ConnectToRemoteObject	Integer	Associates an OLE object with a PowerBuilder OLEObject variable and starts the server application
DisconnectObject	Integer	Releases all objects previously connected
GetAutomationNativePointer	Integer	Returns a pointer to the underlying OLE object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
IsAlive	Boolean	Determines whether a server object is defunct
PostEvent	Boolean	Adds an event to the end of the message queue of the user object
ReleaseAutomationNativePointer	Integer	Releases the pointer to the underlying OLE object
SetAutomationLocale	Integer	Sets the language locale to be used for automation programming on the object
SetAutomationPointer	Integer	Sets the automation pointer of an OLEObject object to the value of the automation pointer of another object
SetAutomationTimeout	Integer	Sets the number of milliseconds that a PowerBuilder client waits before canceling an OLE procedure call to the server
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event
TypeOf	Object	Returns the type of the user object

OLEStorage object

The OLEStorage object acts as a proxy for an open OLE storage.

You can customize your own version of the OLEStorage object by defining a class user object inherited from the built-in OLEStorage object.

For more information about creating a custom OLEStorage object, see the chapter on user objects in the *PowerBuilder Users Guide*.

For more information about using the OLEStorage object in an application, see *Application Techniques*.

Properties

OLEStorage property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DocumentName	String	Specifies the name of the storage currently open. For a root level storage, this is the name of the file containing the storage. For substorages, this is the member name of the substorage.

Events

OLEStorage event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.

Functions

OLEStorage function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object
Clear	Integer	Releases any storage previously opened
Close	Integer	Saves the storage and any controls and streams open on the storage, commits the changes, then releases the storage. (Same as calling Save, then Clear.)
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
MemberDelete	Integer	Deletes the specified member
MemberExists	Integer	Specifies whether the specified member exists
MemberRename	Integer	Renames the specified member

OLEStorage function	Datatype returned	Description
Open	Integer	Opens the storage from a file or opens a substorage within the specified storage object
PostEvent	Boolean	Adds an event to the end of the message queue of the user object
Save	Integer	Saves the storage and any controls open on that storage and commits the changes
SaveAs	Integer	Copies the storage and any controls open on that new storage to a new file or substorage, commits the changes, then releases original storage
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event
TypeOf	Object	Returns the type of the user object

OLEStream object

The OLEStream object acts as a proxy for an OLE stream.

You can customize your own version of the OLEStream object by defining a class user object inherited from the built-in OLEStream object.

For more information about creating a custom OLEStream object, see the chapter on user objects in the *PowerBuilder Users Guide*.

For more information about using the OLEStream object in an application, see *Application Techniques*.

Properties

OLEStream property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Name	String	Specifies the member name of the stream within its parent storage.
Storage	OMStorage	(Read-only) Specifies the storage containing the stream. Streams are always opened from within an OLEStorage object.

Events

OLEStream event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.

Functions

OLEStream function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object.
Close	Integer	Releases any stream previously opened.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Length	Integer	Obtains the length of the open stream.
Open	Integer	Opens the specified stream from the storage.
PostEvent	Boolean	Adds an event to the end of the message queue of the user object.
Read	Integer	Reads from the stream.
Seek	Integer	Moves within the stream.
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event.
TypeOf	Object	Returns the type of the user object.
Write	Long	Writes to the stream.

OLETxnObject object

The OLETxnObject object provides explicit control of MTS transactions to PowerBuilder clients using the SetComplete and SetAbort functions. OLETxnObject inherits from the OLEObject object.

Properties

OLETxnObject property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Handle	Objhandle	Internal use only

Events

OLETxnObject event	Occurs
Constructor	When the object is created.
Destructor	When the object is destroyed.
Error	When an error is found in a data or property expression for an external object.
ExternalException	When the evaluation of an expression involving properties of an external object causes an error. This type of event occurs before the Error event.

Functions

OLETxnObject function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object.
ConnectToNewObject	Integer	Creates a new instance of the class within the transaction context it is associated with and connects to it.
ConnectToNewRemoteObject	Integer	Not used.
ConnectToObject	Integer	Not used.
ConnectToRemoteObject	Integer	Not used.
DisconnectObject	Integer	Releases all objects previously connected.
GetAutomationNativePointer	Integer	Returns a pointer to the underlying OLE object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
IsAlive	Boolean	Determines whether a server object is defunct.
PostEvent	Boolean	Adds an event to the end of the message queue of the user object.
ReleaseAutomationNativePointer	Integer	Releases the pointer to the underlying OLE object.
SetAbort	Integer	Aborts the current transaction.

OLETxnObject function	Datatype returned	Description
SetAutomationLocale	Integer	Sets the language locale to be used for automation programming on the object.
SetAutomationPointer	Integer	Sets the automation pointer of the OLETxnObject object to the value of the automation pointer of another object.
SetAutomationTimeout	Integer	Sets the number of milliseconds that a PowerBuilder client waits before canceling an OLE procedure call to the server.
SetComplete	Integer	Attempts to commit the current transaction.
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event.
TypeOf	Object	Returns the type of the user object.

Oval control

An oval is a filled or outlined round or elliptical drawing object that you typically use for design effects (for example, you can put a CommandButton or a picture in an oval). The grouping does not affect the behavior of the controls in the oval.

Properties

Oval property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
FillColor	Long	Specifies the numeric value of the color used to fill the control: 2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.

Oval property	Datatype	Description
FillPattern	FillPattern (enumerated)	Specifies the hatch pattern used to fill the control. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower right.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
LineColor	Long	Specifies the numeric value of the line color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
LineStyle	LineStyle (enumerated)	Specifies the style of the line used to draw the control. Values are: Continuous! Dash! DashDot! DashDotDot! Dot! Transparent!
LineThickness	Integer	Specifies the thickness of the line used to draw the control, in PowerBuilder units. If LineThickness is greater than one pixel (about four PowerBuilder units), the LineStyle is Continuous!.
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

Oval event	Occurs
Constructor	When the control is created
Destructor	When the control is destroyed

Functions

Oval function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Resize	Integer	Changes the size of the control
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

Picture control

Picture controls can contain images in the following formats:

- Bitmaps, with *.BMP* or *.RLE* extensions
- Windows metafiles, with the *.WMF* extension
- GIF or animated GIF files, with the *.GIF* extension
- JPEG files, with *.JPEG* or *.JPG* extensions
- Portable Networks Graphics, with *.PNG* extension

You can create the image in another application or use a scanner to create it.

Properties

Picture property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.

Picture property	Datatype	Description
FocusRectangle	Boolean	Specifies whether a dotted rectangle (focus rectangle) frames the picture when it has focus. Values are: TRUE – Control is framed when it has focus. FALSE – Control is not framed when it has focus.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Invert	Boolean	Specifies whether the control displays with its colors inverted. Values are: TRUE – Colors are inverted. FALSE – Colors are not inverted.
Map3DColors	Boolean	Specifies whether the system 3D colors are mapped to the control. Values are: TRUE – Colors are mapped. FALSE – Colors are not mapped.
OriginalSize	Boolean	Specifies whether the width and height properties of a bitmap image (picture) are set to their original values. Values are: TRUE – Width and height are set to original values. FALSE – Existing width and height are not changed. In the Window painter, setting OriginalSize to true overrides the existing width and height.
PictureName	String	Specifies the name of the file that contains the picture.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTipText	Long	Specifies a PowerTip for the control.
TabOrder	Integer	Specifies the tab value of the picture within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (distance from the top of the window), in PowerBuilder units.

Events

Picture event	Occurs
Clicked	When the control is clicked (selected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

Picture function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
Draw	Integer	Draws a picture in the parent window at a specified location.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SetFocus	Integer	Sets focus to the control.

Picture function	Datatype returned	Description
SetPicture	Integer	Constructs a new bitmap for the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

PictureButton control

A PictureButton displays a picture and, like a CommandButton, is used to carry out an action. For example, you can use a button with a picture of a file to save a file, or a button with a picture of a stop sign to cancel a requested deletion.

The picture image can be in the following formats:

- Bitmaps, with *.BMP* or *.RLE* extensions
- Windows metafiles, with the *.WMF* extension
- GIF or animated GIF files, with the *.GIF* extension
- JPEG files, with *.JPEG* or *.JPG* extensions
- Portable Networks Graphics, with *.PNG* extension

Properties

PictureButton property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.

PictureButton property	Datatype	Description
BackColor	Long	Specifies the numerical value of the background color of the window. Values are -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window.
Cancel	Boolean	Specifies whether the control acts as the Cancel button (the Cancel button receives a Clicked event if the user presses Esc). Values are: TRUE – Control acts as Cancel button. FALSE – Control does not act as Cancel button.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Default	Boolean	Specifies whether the control is the default PictureButton (the default PictureButton has a thick border and receives a Clicked event if the user presses Enter without selecting an control). Values are: TRUE – Control is default PictureButton. FALSE – Control is not default PictureButton. **Editable controls** Default behavior can be affected by editable controls on the window. For more information, see the *PowerBuilder Users Guide*.
DisabledName	String	Specifies the name of the picture (bitmap image) that displays when the control is disabled. If the string has no extension, PowerBuilder adds an appropriate extension.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.

PictureButton property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FlatStyle	Boolean	Specifies that the edge of the button displays only when the mouse hovers over it. This is the button style used in the Microsoft Rebar (coolbar) control. Values are: TRUE – Button has a flat appearance. FALSE – Button does not have a flat appearance.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HTextAlign	Alignment (enumerated)	Specifies how the text in the control is aligned. Values are: Center! Justify! Left! Right!
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Map3DColors	Boolean	Specifies whether the system 3D colors are mapped to the control. Values are: TRUE – Colors are mapped. FALSE – Colors are not mapped.

PictureButton property	Datatype	Description
OriginalSize	Boolean	Specifies whether the width and height properties of a bitmap image (picture) are set to their original values. Values are: TRUE – Width and height are set to original values. FALSE – Existing width and height are not changed to original values. In the Window painter, setting OriginalSize to true overrides the existing width and height.
PictureName	String	Specifies the name of the file that contains the picture.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTipText	Long	Specifies a PowerTip for the control.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextColor	Long	Specifies the numeric value of the text color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property applies only when the UserObject is a tab page.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VTextAlign	VTextAlign (enumerated)	Specifies how the text in the control is aligned. Values are: Bottom! MultiLine! Top! VCenter! All these values except MultiLine! assume there is only one line of text.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.

PictureButton property	Datatype	Description
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

PictureButton event	Occurs
Clicked	When the control is clicked
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

PictureButton function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.

PictureButton function	Datatype returned	Description
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SetFocus	Integer	Sets focus to the specified control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

PictureHyperLink control

The PictureHyperLink control is a descendant of the Picture control. The URL property of the PictureHyperLink control enables you to provide a hot link to a Web page. When the user clicks the control, the user's Web browser opens to display the page you specify.

PictureHyperLink controls can contain images in the following formats:

- Bitmaps, with *.BMP* or *.RLE* extensions
- Windows metafiles, with the *.WMF* extension
- GIF or animated GIF files, with the *.GIF* extension
- JPEG files, with *.JPEG* or *.JPG* extensions
- Portable Networks Graphics, with *.PNG* extension

Usage note

If you know that your users have Web browsers that support URL completion, you can enter a partial address, such as:

```
appeon.com
```

You can, of course, enter a complete address, such as:

http://www.appeon.com

Properties

PictureHyperLink property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.

PictureHyperLink property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FocusRectangle	Boolean	Specifies whether a dotted rectangle (focus rectangle) frames the picture when it has focus. Values are: TRUE – Control will be framed when it has focus. FALSE – Control will not be framed when it has focus.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Invert	Boolean	Specifies whether the control displays with its colors inverted. Values are: TRUE – Colors are inverted. FALSE – Colors are not inverted.
Map3DColors	Boolean	Specifies whether the system 3D colors are mapped to the control. Values are: TRUE – Colors are mapped. FALSE – Colors are not mapped.
OriginalSize	Boolean	Specifies whether the width and height properties of a bitmap image (picture) are set to their original values. Values are: TRUE – Width and height set to original values. FALSE – Existing width and height not changed. In the Window painter, setting OriginalSize to true overrides the existing width and height.
PictureName	String	Specifies the name of the file that contains the picture. The file extension *BMP*, *RLE*, *WMF*, *JPG*, *JPEG*, or *GIF* is required.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTipText	Long	Specifies a PowerTip for the control.
TabOrder	Integer	Specifies the tab value of the picture within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
URL	String	Specifies the URL to open in the user's Web browser when the picture is clicked, provided no Clicked event is coded. The status text displays the URL when the mouse passes over the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.

PictureHyperLink property	Datatype	Description
X	Integer	Specifies the X position (distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (distance from the top of the window), in PowerBuilder units.

Events

PictureHyperLink event	Occurs
Clicked	When the control is clicked (selected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

PictureHyperLink function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
Draw	Integer	Draws a picture in the parent window at a specified location.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.

PictureHyperLink function	Datatype returned	Description
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SetFocus	Integer	Sets focus to the control.
SetPicture	Integer	Constructs a new bitmap for the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

PictureListBox control

A PictureListBox displays available options or values, which can include pictures. If more options or values exist than can display in the PictureListBox at one time or the text exceeds the width of the PictureListBox, the PictureListBox has one or two (vertical or horizontal) scroll bars.

PictureListBox controls can contain images in the following formats:

- Bitmaps, with the *.BMP* extension
- Icons, with the *.CUR* extension
- Cursors, with the *.ICO* extension
- GIF files, with the *.GIF* extension, but not animated GIF files
- JPEG files, with *.JPEG* or *.JPG* extensions
- Portable Networks Graphics, with *.PNG* extension

Properties

PictureListBox property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisableNoScroll	Boolean	Specifies behavior of scroll bar. Values are: TRUE – The scroll bar is always visible, but is disabled when all the items can be accessed without it. FALSE – The scroll bar is displayed only if it is necessary (based on the number of items and the height of the list box).
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.

PictureListBox property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
ExtendedSelect	Boolean	Specifies whether users can select multiple items in the list box at one time. Values are: TRUE – Users can select multiple items by clicking on an item and dragging the mouse up or down to select items; using Click or Shift+ Click to select a sequential group of items; or using Ctrl+ Click on multiple items. FALSE – Users cannot select multiple items. **Used with MultiSelect** The MultiSelect property allows users to select multiple items in a list box by simply clicking on the items. If MultiSelect = true and ExtendedSelect = true, then the behavior of ExtendedSelect takes precedence.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!

PictureListBox property	Datatype	Description
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays. Values are: TRUE – Horizontal scroll bar displays. FALSE – Horizontal scroll bar does not display.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Item[ ]	String	Specifies the items in the control. Not updated after initialization.
ItemPictureIndex[ ]	Integer	Specifies the picture index for each item in the Item property array. Not updated after initialization.
MultiSelect	Boolean	Specifies whether users can select multiple items in the PictureListBox at one time. Values are: TRUE – Users can select multiple items. FALSE – Users cannot select multiple items. **Used with ExtendedSelect** The MultiSelect property allows users to select multiple items in a list box by simply clicking on the items. If MultiSelect = true and ExtendedSelect = true, then the behavior of ExtendedSelect takes precedence.
PictureHeight	Integer	Specifies height of the picture, in pixels. In a script, this property can be set only when there are no images in the image list. If the value is 0 at the time the first image is added, the size of that image is used to set the size of the rest of the images added.
PictureMaskColor	Long	Specifies the numeric value of the color to be used to mask user-defined bitmaps added through the initial picture array or with the AddPicture function. System-defined bitmaps know their mask color so this color is ignored. This value is used when a picture is added, and therefore can be changed between AddPicture calls. Values can be: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
PictureName[ ]	String	Specifies the names of the files containing the pictures added during initialization. The file extension *BMP*, *ICO*, *GIF*, *JPG* or *JPEG* is required. Not updated after initialization.

PictureListBox property	Datatype	Description
PictureWidth	Integer	Specifies width of the picture, in pixels. In a script, this property can be set only when there are no images in the image list. If the value is 0 at the time the first image is added, the size of that image is used to set the size of the rest of the images added.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
Sorted	Boolean	Specifies whether the items in the PictureListBox are sorted. Values are: TRUE – Items are sorted. FALSE – Items are not sorted.
TabOrder	Integer	Specifies the tab value of the control (0 means the user cannot tab to the control).
TabStop[ ]	Integer array	Specifies the positions of the tab stops in the PictureListBox. The tab stops are in character positions, and the tab stop delimiter is a space. If you assign a value to only the first tab stop, TabStop[1], the tab stops are equally spaced using the number of character positions specified for the first tab stop. If more than one tab stop is entered, tab stops are located in the positions specified. You can define 16 tab stops in the control; the default array is TabStop[8], with a tab stop every eight character positions.
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.

PictureListBox property	Datatype	Description
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar is displayed on the right of the PictureListBox. Values are: TRUE – Vertical scroll bar is displayed. FALSE – Vertical scroll bar is not displayed.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

PictureListBox event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Control message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control
SelectionChanged	When an item in the control is selected

Functions

PictureListBox function	Datatype returned	Description
AddItem	Integer	Adds a new item to the end of the PictureListBox. If the Sorted property of the control is true, the items are sorted again after the item is added. The AddItem function does not update the Item[ ] or ItemPicture[ ] properties of this control.
AddPicture	Integer	Adds the bitmap, icon, or cursor file to the main image list. Does not update PictureName[ ].
ClassName	String	Returns the name assigned to the control.
DeleteItem	Integer	Deletes the item indicated by the index from the PictureListBox.
DeletePicture	Integer	Deletes the specified picture from the image list. Does not update PictureName[ ].
DeletePictures	Integer	Deletes all the pictures from the image list. Does not update PictureName[ ].
DirList	Boolean	Populates the PictureListBox with a list of the files of the specified type that match the specified file pattern.
DirSelect	Boolean	Returns the current selection for the control and puts it in the specified variable.
Drag	Integer	Starts or ends the dragging of a control.
FindItem	Integer	Finds the first item in the PictureListBox (after the specified index) that begins with the specified string.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
InsertItem	Integer	Adds a new item to the PictureListBox before the item indicated by the index. If the Sorted property of the control is true, the items are sorted again after the item is added.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for the control.
Print	Integer	Prints the control.
Reset	Integer	Removes all items from the control.
Resize	Integer	Changes the size of the control.

PictureListBox function	Datatype returned	Description
SelectedIndex	Integer	Returns the index of the item in the PictureListBox that is currently selected. If more than one item is selected, it returns the index of the first selected item.
SelectedItem	String	Returns the text of the first selected item.
SelectItem	Integer	Finds and highlights an item in the control. Use Syntax 1 when you know the text of the item but not its position. Use Syntax 2 when you know the position of the item in the control's list or you want to clear the current selection. SelectItem has no effect on a PictureListBox whose MultiSelect property is true. Instead, use SetState to select items without affecting the selected state of other items in the list.
SetFocus	Integer	Sets focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
SetState	Integer	Sets the state (highlighted or not highlighted) of the item indicated by the specified index. SetState works only for multiselect controls (that is, those for which the MultiSelect property is true).
SetTop	Integer	Scrolls the items in the control so that the item indicated by the specified index is at the top of the control.
Show	Integer	Makes the control visible.
State	Integer	Returns 1 if the item specified by the specified index is selected (highlighted) and 0 if the item is not selected.
Text	String	Returns the text of the item in the control that is identified by the specified index.
Top	Integer	Returns the index number of the item currently at the top of the control.
TotalItems	Integer	Returns the total number of items in the control.
TotalSelected	Integer	Returns the total number of items selected in the control.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

Pipeline object

A Pipeline system object is used to manage a data pipeline during execution. You use a Pipeline object by defining a standard class user object inherited from the built-in Pipeline object in the User Object painter. You can then access the Pipeline events by writing scripts that contain code for the events.

For more information about piping data, see *Application Techniques*.

Properties

Pipeline property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
DataObject	String	The name of the pipeline object (the object created in the Data Pipeline painter, but that you must assign dynamically at runtime)
RowsInError	Long	The number of rows the pipeline found in error (for example, rows containing a duplicate key)
RowsRead	Long	The number of rows read by the pipeline
RowsWritten	Long	The number of rows written by the pipeline
Syntax	String	The syntax used to create the pipeline object (the object created in the Data Pipeline painter)

Events

Pipeline event	Occurs
Constructor	When the user object is created.
Destructor	When the user object is destroyed.
PipeEnd	When Start or Repair is completed.
PipeMeter	After each block of rows is read or written. The Commit factor specified for the pipeline determines the size of each block.
PipeStart	When a Start or Repair is started.

Functions

Pipeline function	Datatype returned	Description
Cancel	Integer	Stops execution of a pipeline.
ClassName	String	Returns the name assigned to the user object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	Power Object	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds an event to the end of the message queue of the user object.
Repair	Integer	Updates the target database with corrections that have been made in the pipeline user object's Error DataWindow.
Start	Integer	Executes a pipeline.
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event.
TypeOf	Object	Returns the type of the user object.

ProfileCall object

The ProfileCall object provides information about the calls in the performance analysis model, including information about the called routine and the calling routine, the number of times the call was made, and the elapsed time (in seconds). You use the ProfileCall object in conjunction with the ProfileRoutine and Profiling objects.

The ProfileCall object has no events.

Properties

ProfileCall property	Datatype	Description
AbsoluteSelfTime	Decimal	The time (in seconds) spent in the called routine.
AbsoluteTotalTime	Decimal	The time (in seconds) spent in the called routine and in subsequent called routines.
CalledRoutine	ProfileRoutine	An object of datatype ProfileRoutine containing the destination of the call.

ProfileCall property	Datatype	Description
CallingLine	ProfileLine	An object of datatype ProfileLine containing the initiating line of the call. If the call object represents an aggregation of multiple calls from a routine, an invalid object is returned.
CallingRoutine	ProfileRoutine	An object of datatype ProfileRoutine containing the routine that initiated the call.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
HitCount	Long	The number of times the calling routine called the called routine.
PercentCalleeSelfTime	Double	AbsoluteSelfTime as a percentage of the total time (in seconds) the calling routine was active.
PercentCalleeTotalTime	Double	AbsoluteTotalTime as a percentage of the total time (in seconds) the calling routine was active.
PercentCallerTotalTime	Double	The total time (in seconds) spent in the calling routing as a percentage of the total time the calling routine was active.

Functions

ProfileCall function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

ProfileClass object

The ProfileClass object provides information about the classes in the performance analysis model, including the routines that exist within a class. You use the ProfileClass object in conjunction with the Profiling object.

The ProfileClass object has no events.

Properties

ProfileClass property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
LibraryName	String	The name of the library that contains the class. The value is " " for system classes and embedded SQL statements.
Name	String	The name of the class or the string `Embedded SQL` to represent all embedded SQL activities. Nested classes (like controls on a window) have a name of the form *class name>embedded class name*.

Functions

ProfileClass function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	Power Object	Returns a reference to the name of the parent object
RoutineList	ErrorReturn (enumerated)	Provides a list of the routines (defined as ProfileRoutine objects) that exist in the model within a class
TypeOf	Object (enumerated)	Returns the type of the object

ProfileLine object

The ProfileLine object provides information about the lines in each routine in the performance analysis model, including the number of times the line was hit, any calls made from the line, and the time (in seconds) spent on the line and in any called functions. You use the ProfileLine object in conjunction with the ProfileRoutine and Profiling objects.

The ProfileLine object has no events.

Properties

ProfileLine property	Datatype	Description
AbsoluteSelfTime	Decimal	The time (in seconds) spent on this line itself. If the line executed more than once, this is the total time spent on the line.
AbsoluteTotalTime	Decimal	The time (in seconds) spent on this line and on lines called from this line. If the line executed more than once, this is the total time spent on the line and on called lines.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
HitCount	Long	The number of times the line was called.
LineNumber	Long	The line number. Line 0 is a special line that represents the time (in seconds) taken to initialize the local variables (including calling constructors for autoinstantiated objects).
MaxSelfTime	Decimal	The longest time (in seconds) spent just on this line. If the line executed only once, this is the same as AbsoluteSelfTime.
MaxTotalTime	Decimal	The longest time (in seconds) spent on this line and on called lines. If the line executed only once, this is the same as AbsoluteTotalTime.
MinSelfTime	Decimal	The shortest time (in seconds) spent just on this line. If the line executed only once, this is the same as AbsoluteSelfTime.
MinTotalTime	Decimal	The shortest time (in seconds) spent on this line and on called lines. If the line executed only once, this is the same as AbsoluteTotalTime.
PercentSelfTime	Double	AbsoluteSelfTime as a percentage of the total time (in seconds) tracing was active.
PercentTotalTime	Double	AbsoluteTotalTime as a percentage of the total time (in seconds) tracing was active.
Routine	ProfileRoutine	The routine that the line is in.

Functions

ProfileLine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	Power Object	Returns a reference to the name of the parent object

ProfileLine function	Datatype returned	Description
OutgoingCallList	ErrorReturn (enumerated)	Provides a list of the calls (defined as ProfileCall objects) to other routines from a line
TypeOf	Object (enumerated)	Returns the type of the object

ProfileRoutine object

The ProfileRoutine object provides information about the routines in the performance analysis model. It includes the time (in seconds) spent in the routine, any called routines, the number of times each routine was called, and the class to which the routine belongs. You use the ProfileRoutine object in conjunction with the Profiling and ProfileCall or ProfileLine objects.

The ProfileRoutine object has no events.

Properties

ProfileRoutine property	Datatype	Description
AbsoluteSelfTime	Decimal	The time (in seconds) spent in this routine. If the routine executed more than once, this is the total time spent in the routine.
AbsoluteTotalTime	Decimal	The time (in seconds) spent in this routine and in routines called from this routine. If the routine executed more than once, this is the total time spent in the routine and in called routines.
Class	ProfileClass	The class the routine is in. For embedded SQL activities, the value is `Embedded SQL`. For global and system functions, the value is an invalid object.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
HitCount	Long	The number of times this routine was called or the number of objects created or destroyed.

ProfileRoutine property	Datatype	Description
Kind	ProfileRoutine Kind (enumerated)	The kind of routine node. Values are: RoutineESQL! – Represents an embedded SQL statement. RoutineEvent! – Represents an event. RoutineFunction! – Represents a function. RoutineGarbageCollection! – Represents a garbage collection phase. RoutineObjectCreation! – Represents object creation. RoutineObjectDestruction! – Represents object destruction. RoutineRoot! – Represents the windowing system.
MaxSelfTime	Decimal	The longest time (in seconds) spent in the routine itself. If the routine executed only once, this is the same as AbsoluteSelfTime.
MaxTotalTime	Decimal	The longest time (in seconds) spent in the routine and in called routines. If the routine executed only once, this is the same as AbsoluteTotalTime.
MinSelfTime	Decimal	The shortest time (in seconds) spent in the routine itself. If the routine executed only once, this is the same as AbsoluteSelfTime.
MinTotalTime	Decimal	The shortest time (in seconds) spent in the routine and in called routines. If the routine executed only once, this is the same as AbsoluteTotalTime.
Name	String	The name of the routine including the argument datatypes and return value. For embedded SQL activities, the value is the name of the statement (for example, SELECT). For object creation or destruction, the value is Object Create or Object Destroy.
PercentSelfTime	Double	AbsoluteSelfTime as a percentage of the total time (in seconds) tracing was active.
PercentTotalTime	Double	AbsoluteTotalTime as a percentage of the total time (in seconds) tracing was active.

Functions

ProfileRoutine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service

ProfileRoutine function	Datatype returned	Description
GetParent	Power Object	Returns a reference to the name of the parent object
IncomingCallList	ErrorReturn (enumerated)	Provides a list of the callers (defined as ProfileCall objects) of this routine
LineList	ErrorReturn (enumerated)	Provides a list, in line order, of the lines (defined as ProfileLine objects) in the routine
OutgoingCallList	ErrorReturn (enumerated)	Provides a list of the calls (defined as ProfileCall objects) to other routines from within this routine
TypeOf	Object (enumerated)	Returns the type of the object

Profiling object

The Profiling object is used to analyze the performance of a PowerBuilder application. It provides a performance analysis model listing all the routines (both functions and events) logged in a given trace file. It includes the functions you call to name the trace file to be analyzed, build the model, and list the classes and routines included in the model. You use the Profiling object in conjunction with the ProfileCall, ProfileClass, ProfileLine, and ProfileRoutine objects.

The Profiling object has no events.

Properties

Profiling property	Datatype	Description
ApplicationName	String	The name of the application used to generate the trace file.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CollectionTime	Decimal	The amount of time taken (in seconds) by the collection of trace data. This time has already been accounted for in the timestamps from the trace file (that is, the proper amount of time is subtracted from the timestamps before they are put in the trace file). If no model has been created, NULL is returned.

Profiling property	Datatype	Description
NumberOfActivities	Long	The total number of activities that exist in the trace file. The value is 0 if this property is called before the trace file name is set.
TraceFileName	String	The name of the trace file to use to build the model. The value is an empty string if the name has not been successfully set.

Functions

Profiling function	Datatype returned	Description
BuildModel	ErrorReturn (enumerated)	Builds a performance analysis model based on the previously specified trace file
ClassList	ErrorReturn (enumerated)	Provides a list of the classes (defined as ProfileClass objects) included in the model
ClassName	String	Returns the name assigned to the object
DestroyModel	ErrorReturn (enumerated)	Destroys the current performance analysis model
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	Power Object	Returns a reference to the name of the parent object
RoutineList	ErrorReturn (enumerated)	Provides a list of the routines (defined as ProfileRoutine objects) included in the model
SetTraceFileName	ErrorReturn	Indicates the name of the trace file to use for analysis and validates the header format
SystemRoutine	ProfileRoutine	Provides the routine node (defined as a ProfileRoutine object) representing the system root
TypeOf	Object (enumerated)	Returns the type of the object

RadioButton control

A RadioButton is a small round button that is used to turn an option on and off. When the option is on, the button has a dark center. When the option is off, the center is blank.

In a GroupBox
RadioButtons are often grouped in a GroupBox. In this case, the user can select only one button in the group, and the group usually has a default button.

Properties

RadioButton property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Automatic	Boolean	Specifies whether the control becomes dark when it is clicked. Values are: TRUE – Control becomes dark when clicked. FALSE – Control does not become dark when clicked.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleLowered! StyleRaised!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
Checked	Boolean	Specifies whether the item is selected (the center is dark). Values are: TRUE – Control is selected. FALSE – Control is not selected.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

RadioButton property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control is enabled. FALSE – Control is not enabled.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Helv or Courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.

RadioButton property	Datatype	Description
LeftText	Boolean	Specifies whether the text displays to the left of the control. Values are: TRUE – Text displays to the left. FALSE – Text does not display to the left.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control (0 means the user cannot tab to the control). In a GroupBox, the up and down arrow keys are used to move among RadioButtons in a specified sequence. To permit tabbing in a GroupBox, change the tab value of the GroupBox to 0 and assign nonzero tab values to the RadioButtons (the default tab value for the RadioButtons in a GroupBox is 0).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays next to the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

RadioButton event	Occurs
Clicked	When the control is clicked (selected or unselected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

RadioButton function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window

RadioButton function	Datatype returned	Description
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event for the control and executes the script for the event
TypeOf	Object	Returns the type of the control

Rectangle control

A rectangle is a filled or outlined rectangular form within a window and is typically used for design purposes. For example, you can put a CommandButton or a picture in a rectangle, or you can use a rectangle behind and slightly offset from another control to create a shadow effect. When you use a rectangle to group controls, the grouping does not affect the behavior of the controls in the rectangle.

Properties

Rectangle property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
FillColor	Long	Specifies the numeric value of the color used to fill the control: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
FillPattern	FillPattern (enumerated)	Specifies the hatch pattern used to fill the control. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower-right.
Height	Integer	Specifies the height of the control, in PowerBuilder units.

Rectangle property	Datatype	Description
LineColor	Long	Specifies the numeric value of the line color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
LineStyle	LineStyle (enumerated)	Specifies the pattern of the line used to draw the control. Values are: Continuous! Dash! DashDot! DashDotDot! Dot! Transparent!
LineThickness	Integer	Specifies the thickness of the line used to draw the control, in PowerBuilder units. If LineThickness is greater than one pixel (about four PowerBuilder units), the LineStyle is forced to Continuous!
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

Rectangle event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window

Functions

Rectangle function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control

Rectangle function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Resize	Integer	Changes the size of the control
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

ResultSet object

The ResultSet object provides the ability to use ActiveX Data Object (ADO) record sets to return a result set to a client. Use ResultSet objects with the CreateFrom and GenerateResultSet DataStore functions.

Properties

ResultSet property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

ResultSet event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

ResultSet function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

ResultSets object (obsolete)

Obsolete object
ResultSets object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The ResultSets object provides the ability to handle multiple result sets returned from EAServer.

Properties

ResultSets property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
ResultSetList	ResultSet object	An array of ResultSet objects

Events

ResultSets event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

ResultSets function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

RichTextEdit control

A RichTextEdit control contains a document that it displays as formatted text. It can include input fields that are linked to a DataWindow control. When a DataWindow's data is shared with the RichTextEdit control, there is one instance of the document in the control that can be displayed multiple times with different occurrences of row data. Input fields whose names match columns in the DataWindow are filled with data from the current row.

New RichTextEdit control in PowerBuilder 2017

In PowerBuilder 2017, a new RichTextEdit control is implemented. The properties/functions/events are exactly the same for the new control and the old control, except for those specified in the following tables and in the summary table in the What's New document.

By default the new control is used. The new control is applicable to the RichTextEdit control, RichText DataWindow Object, and the RichText edit style column.You can switch the control in the application's Additional Properties window.

Using with animation features

RichTextEdit controls may not paint correctly when you use animation features.

Properties

RichTextEdit property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the accelerator key you want to assign as the accelerator for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BackColor	Long	Specifies the numeric value of the background color of the text editing area of the RichTextEdit. Values are –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. **Difference between the new control and the old control:** For the new control, if you set a negative value for the BackColor property, the returned value will be 0 and the color will be black. For the old control, if you set a negative value for the BackColor property, the returned value will be 16777215 and the color will be white.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Has a border. FALSE – Does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BottomMargin	Long	Specifies the width of the bottom margin on the printed page. **Difference between the new control and the old control:** For the new control, if you set a negative value, the returned value will be 0. For the old control, if you set a negative value, the returned value will be the negative value. This difference also exists in the other margin properties, such as TopMargin, LeftMargin, and RightMargin.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Moves it to the top. FALSE – Does not move it to the top.

RichTextEdit property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ControlCharsVisible	Boolean	Specifies whether control characters (carriage returns, spaces, and tabs) are visible. Values are: TRUE – Control characters are visible. FALSE – Control characters are hidden.
DisplayOnly	Boolean	Specifies whether users can make changes to the content. Values are: TRUE – The content, including text and input files, is protected (the user cannot edit it). FALSE – The user can edit the content.
DocumentName	String	Specifies the name that displays in the print queue when the user prints the contents of the control.
DragAuto	Boolean	(**Obsolete**) This property is no longer supported by the RichTextEdit control since PowerBuilder 12.6. Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Can be selected. FALSE – Cannot be selected.
FaceName	String	Specifies the typeface used for text in the control.
FontCharSet	FontCharSet (enumerated)	Specifies the character set for text in the control.
FontFamily	FontFamily (enumerated)	Specifies the font family for text in the control.
FontPitch	FontPitch (enumerated)	Specifies the spacing of the font used for text in the control.

RichTextEdit property	Datatype	Description
HeaderFooter	Boolean	Specifies whether the RichTextEdit control has a header/footer section. This property must be set in the painter and cannot be changed during execution. Values are: TRUE – The control has a header/footer section. FALSE – The control does not have a header/footer section. If a document has a header or footer and is opened in a control that does not support a header/footer section, then header/footer information in the document is ignored. If the document is then saved in the same file, the header/footer information is lost.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether the RichTextEdit control has a horizontal scroll bar. Values are: TRUE – A scroll bar displays. FALSE – A scroll bar does not display.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
InputFieldBackColor	Long	Specifies default background color for all input fields: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. **Limitation with the new control:** For the new control, this property will not take effect until you save the data into a PDF file or print the data; which means, when you preview the UI in the design view or when you run the UI, you will always see the background color is gray, only when you run the UI and save the data to a PDF file or print the data, you will see the background color is changed to what you set.
InputFieldNamesVisible	Boolean	Specifies whether input field names are displayed in input fields, rather than the input field values. Values are: TRUE – Input fields display their names. FALSE – Input fields display their data.
InputFieldsVisible	Boolean	Specifies whether input fields are visible. Values are: TRUE – Input fields are visible. FALSE – Input fields are hidden.
Italic	Boolean	Specifies default text formatting. Values are: TRUE – Text displays in italic format. FALSE – Text displays in standard format. Maintained for forward compatibility with PowerBuilder .NET. In PowerBuilder applications, use the check box on the Font tab of the Properties view to set this property at initialization.
LeftMargin	Long	Specifies the width of the left margin on the printed page.

RichTextEdit property	Datatype	Description
Modified	Boolean	Specifies whether the document has been modified since it was opened or last saved. Modified is the control's "dirty" flag, indicating that the document is in an unsaved state. Values are: TRUE – The document has been modified. FALSE – The document has not been modified. When the first change is made to a newly opened or saved document, PowerBuilder sets the Modified attribute and triggers the Modified event.
PaperHeight	Long	Specifies the value for the display height of pages in the control.
PaperOrientation	PaperOrientation (enumerated)	Specifies the page orientation in the control.
PaperWidth	Long	Specifies the value for the display width of pages in the control.
PicturesAsFrame	Boolean	(**Obsolete**) This property is no longer supported by the RichTextEdit control since PowerBuilder 12.6. When the value is true, graphics (bitmaps) are displayed as frames.
Pointer	String	Specifies the name of the stock pointer of the file containing the pointer that is used for the control.
PopMenu	Boolean	Specifies whether the user has access to a pop-up menu by clicking the right mouse button on the control. The menu allows the user to cut and paste, insert a file, and select formatting options. Values are: TRUE – Pop-up menu is enabled. FALSE – Pop-up menu is disabled.
Resizable	Boolean	Specifies whether the user can resize the control. Values are: TRUE – Control is resizable. FALSE – Control is not resizable.
RightMargin	Long	Specifies the width of the right margin on the printed page.
RulerBar	Boolean	Specifies whether a ruler bar is visible above the editing area. If visible, the user can use it to see measurements while setting tabs and margins on the tab bar (see the TabBar property). Values are: TRUE – Ruler bar is visible. FALSE – Ruler bar is hidden. If the RichTextEdit pop-up menu is enabled, the user can use it to turn ruler bar display on and off (see the PopMenu property).
SelectedStartPos	Long	Specifies the starting position in a selected text string. Typically, you use this property to set the starting position of a selected text string to the first letter of a word flagged by a supported ActiveX spell checker control.

RichTextEdit property	Datatype	Description
SelectedTextLength	Long	Specifies the length of text you want to highlight in a selected text string. Typically you use this property to obtain the length of a misspelled word that is flagged after passing the selected text string to a supported ActiveX spell checker control.
StatusBar	Boolean	Specifies whether a status bar is visible below the editing area. Values are: TRUE – Status bar is visible. FALSE – Status bar is hidden. If the pop-up menu is enabled, the user can use it to turn the status bar display on and off (see the PopMenu property).
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control). Note that because a RichTextEdit allows tabs within its content, the user cannot tab away from the control.
Tag	String	Specifies the tag value assigned to the control.
TextSize	Integer	Specifies the point size of text in the control.
ToolBar	Boolean	Specifies whether a tool bar for formatting text is visible above the editing area. Values are: TRUE – Tool bar is visible. FALSE – Tool bar is not visible. If the pop-up menu is enabled, the user can use it to turn tool bar display on and off (see the PopMenu property).
TopMargin	Long	Specifies the width of the top margin on the printed page.
Underline	Boolean	Specifies default text formatting. Values are: TRUE – Text displays with underlines. FALSE – Text displays without underlines. Maintained for forward compatibility with PowerBuilder .NET. In PowerBuilder applications, use the check box on the Font tab of the Properties view to set this property at initialization.
Visible	Boolean	Specifies whether the control is visible: TRUE – Control is visible. FALSE – Control is not visible.
VScrollBar	Boolean	Specifies whether the RichTextEdit control has a vertical scroll bar. Values are: TRUE – A scroll bar displays. FALSE – A scroll bar does not display.

RichTextEdit property	Datatype	Description
Weight	Integer	Specifies default text weight formatting for the control. Maintained for forward compatibility with PowerBuilder .NET. In PowerBuilder applications, use the Bold check box on the Font tab of the Properties view to set bold text weight formatting at initialization.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
WordWrap	Boolean	Determines whether large blocks of text that do not contain spaces wrap automatically to the next line when the line reaches the margin. Values are: TRUE – Automatic word wrap is enabled. FALSE – Automatic word wrap is disabled. Users cannot enter characters beyond the right margin, and must move the cursor to a new line to continue entering text. If an inserted document contains a block of text too large to fit on a line, the nonfitting characters are hidden. **Limitation with the new control:** For the new control, this property is always true (even when it is set to false, it is treated as true).
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

RichTextEdit event	Occurs
Constructor	Immediately before the Open event occurs in the window.
Destructor	Immediately after the Close event occurs in the window.
DoubleClicked	When the user double-clicks anywhere in the RichTextEdit control.
DragDrop	When a dragged control is dropped on the control.
DragEnter	When a dragged control enters the control.
DragLeave	When a dragged control leaves the control.
DragWithin	When a dragged control is within the control.
FileExists	When the document in the RichTextEdit is saved to a file, but a file of the specified name already exists.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.

RichTextEdit event	Occurs
InputFieldSelected	When the user selects an input field by double-clicking it.
Key	When the user presses a key.
LoseFocus	When the control loses focus (becomes inactive).
Modified	When the first change is made to the contents of the RichTextEdit control, and it has not been saved. In the Modified event script, you can set a flag indicating that the document needs to be saved.
MouseDown	When the left or middle mouse button is pressed on the control.
MouseMove	When the mouse has moved within the control.
MouseUp	When the left or middle mouse button is released on the control.
Other	When a Windows message occurs that is not a PowerBuilder event.
PictureSelected	When the user selects a picture by clicking it.
RButtonDown	When the right mouse button is pressed on the control. If the pop-up menu is enabled, this event does not occur.
RButtonUp	When the right mouse button is released on the control.

Functions

RichTextEdit function	Datatype returned	Description
CanUndo	Boolean	Returns true if there is an editing function that can be undone.
ClassName	String	Returns the name assigned to the control.
Clear	Long	Clears selected text (if any) from the control, but does not place it in the clipboard.
ClearAll	Long	Clears all content from the specified control, but does not place it in the clipboard.
Copy	Long	Copies (but does not delete) the selected contents of the RichTextEdit control (if any) to the clipboard.
CopyRTF	String	Copies the selected contents of the RichTextEdit control to a string in rich text format. **Difference between the new control and the old control:** The new control will return a longer string which contains more types of information, compared to the old control.
Cut	Long	Cuts (deletes) the selected contents from the RichTextEdit control (if any) to the clipboard.
DataSource	Integer	Associates a DataWindow with the RichTextEdit control, matching columns with input fields of the same name.
Drag	Integer	Starts or ends dragging of the control.

RichTextEdit function	Datatype returned	Description
Find	Integer	Finds text in the control. **Enhancement of the new control:** For the new control, the Find function can not only find the text string, but also can find the carriage return characters and some special characters.
FindNext	Integer	Finds the next occurrence of text specified with the Find function.
GetAlignment	Alignment	Obtains the alignment of the selected text. **Difference between the new control and the old control:** For the new control, this property will return the alignment of the paragraph where the insertion point is located if more than one paragraph are selected. For the old control, this property will return empty if more than one paragraph with different alignments are selected.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParagraphSetting	Long	Gets the size of the indentation, left margin, or right margin of the paragraph containing the insertion point. The returned value is different between the new control and the old control due to the different units used. **Difference between the new control and the old control:** For the new control, this property will return the settings of the paragraph where the insertion point is located if more than one paragraph are selected. For the old control, this property will return empty if more than one paragraph with different settings are selected.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetSpacing	Spacing	Obtains the line spacing of the selected text. **Difference between the new control and the old control:** For the new control, this property will return the spacing of the paragraph where the insertion point is located if more than one paragraph are selected. For the old control, this property will return empty if more than one paragraph with different spacing are selected.
GetTextColor	Long	Returns the color of the selected text. **Difference between the new control and the old control:** For the new control, this function will return the text color of the first character of the selected text if the selected text contains more than one text colors. For the old control, this property will return empty if the selected text contains more than one text color.

RichTextEdit function	Datatype returned	Description
GetTextStyle	Boolean	Obtains font settings for the selected text. **Difference between the new control and the old control:** For the new control, this function will return the font settings of the first character of the selected text if the selected text contains more than one type of font settings. For the old control, this property will return empty if the selected text contains more than one type of font settings.
Hide	Integer	Makes the control invisible.
InputFieldChangeData	Integer	Modifies the data value for all input fields of the given input field name.
InputFieldCurrentName	String	Gets the name of the selected input field.
InputFieldDeleteCurrent	Integer	Deletes the current occurrence of the selected input field. (That is, this function does not delete all input fields of the same name, only the current occurrence.)
InputFieldGetData	String	Obtains the data in the specified input field.
InputFieldInsert	Integer	Inserts the named input field at the insertion point.
InputFieldLocate	String	Locates an input field. You can find any occurrence of an input field or an input field with a specific name.
InsertDocument	Integer	Inserts the named file in the RichTextEdit control. The file can replace the current contents or be added at the insertion point. The file can be in rich text (RTF), ASCII, HTML, or Word (DOC) format.
InsertPicture	Integer	Inserts the specified bitmap or picture file at the insertion point. The inserted WMF image file will fail to be saved in to a Word document, although it can be saved in to a rich text file.
IsPreview	Boolean	Checks whether the RichTextEdit control is in preview mode.
LineCount	Integer	Returns the total number of lines in the document.
LineLength	Integer	Returns the length of the current line.
Move	Integer	Moves the control to a specified location.
PageCount	Integer	Returns the number of pages in the document.
Paste	Integer	Inserts the contents of the clipboard (if any) at the insertion point in the control.
PasteRTF	Long	Inserts a string at the insertion point when the string contains text in rich text format. You can specify whether to insert the string in the header or footer band, as well as the main body (detail band).
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
Position	Integer	Returns the position of the insertion point or selected text in the control.

RichTextEdit function	Datatype returned	Description
PostEvent	Integer	Adds the specified event to the end of the event queue for the specified object
Preview	Integer	Flips between Preview of the document (where header/footer, page size, and so on, are shown) and normal RTE control display.
Print	Integer	Prints the contents of the control. You can specify a range of pages and other settings.
PrintEx	Integer	Prints the contents of the control. You can specify whether a Print dialog box displays.
ReplaceText	Integer	Replaces selected text with given string. If no text has been selected, insert given string at cursor location. **Limitation with the new control:** In the new control, the text after replaced will take over the settings (such as font name, font size etc.) carried over by the given string.
Resize	Integer	Changes the size of the control.
SaveDocument	Integer	Saves the current document in a file in either rich text (RTF), ASCII, HTML, PDF, or Word (DOC) format. When saving multi-byte characters to a TXT file, use the UTF8 encoding rather than the ANSI encoding. **Limitation with the new control:** In the new control, when saving to HTML, the image will lose some quality and will be saved as individual files separately from the document and the reference to the image file uses the absolute path.
Scroll	Integer	Scrolls the RichTextEdit the specified direction the specified number of lines.
ScrollNextPage	Integer	Scrolls forward to the next page in the RichTextEdit document. If the current page is the last page of a document instance, scrolls to the next instance.
ScrollNextRow	Long	Scrolls the RichTextEdit to the next row and document instance.
ScrollPriorPage	Long	Scrolls back to the prior page in the RichTextEdit document. If the current page is the first page of a document instance, scrolls to the prior instance.
ScrollPriorRow	Long	Scrolls the RichTextEdit to the prior row and document instance.
ScrollToRow	Long	Causes the control to scroll to the specified row and document instance.
SelectedColumn	Integer	Returns the number of the character column just after the insertion point.
SelectedLength	Long	Returns the length of the selected text.
SelectedLine	Long	Returns the number of the line in which the insertion point is currently located within the document instance.

RichTextEdit function	Datatype returned	Description
SelectedPage	Long	Returns the number of the page in which the insertion point is currently located within the document instance. **Limitation with the new control:** For the new control, the SelectedPage function returns the number of the page which is being viewed (rather than where the insertion point is placed).
SelectedStart	Integer	Returns the starting position of the selected text (if any) from the beginning of the line.
SelectedText	String	Returns a string containing the selected text from the control.
SelectText	Long	Selects text in the RichTextEdit control.
SelectTextAll	Integer	Selects all text in the control.
SelectTextLine	Integer	Selects all text in the line in which the insertion point is currently located.
SelectTextWord	Integer	Selects the word in which the insertion point is currently located.
SetAlignment	Integer`	Sets the alignment for the selected paragraphs.
SetFocus	Integer	Sets focus to the RichTextEdit control.
SetParagraphSetting	Integer	Specifies the size of the paragraph's indentation, left margin, or right margin.
SetPosition	Integer	Specifies the control's position in the front-to-back order within the window.
SetRedraw	Integer	Controls automatic redrawing of the control.
SetSpacing	Integer	Sets the line spacing for the selected paragraphs.
SetTextColor	Integer	Sets the color of the selected text.
SetTextStyle	Integer	Sets the font properties of the selected text.
Show	Integer	Makes the RichTextEdit control visible.
ShowHeadFoot	Integer	Allows editing of the header and footer in the RichTextEdit document. **Difference between the new control and the old control:** When the document is in preview mode, the old control returns 1; while the new control returns 1 and closes the preview mode, and when the parameter is true, the insertion point is placed to the header area; when the parameter is false, the insertion point is placed to the detail area.
TextLine	String	Returns the entire text of the line in which the insertion point is currently located.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

RichTextEdit function	Datatype returned	Description
Undo	Integer	Cancels the previous editing function performed in the control, if the editing function can be undone. (Some editing functions cannot be undone.)

RoundRectangle control

A RoundRectangle is a filled or outlined rectangular drawing object with rounded corners that you typically use for design purposes (for example, you can put a CommandButton or a picture in a RoundRectangle). When you use a RoundRectangle to group controls, the grouping does not affect the behavior of the controls in the RoundRectangle.

Properties

RoundRectangle property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CornerHeight	Integer	Specifies the radius of the vertical part of the corners of the control, in PowerBuilder units.
CornerWidth	Integer	Specifies the radius of the horizontal part of the corners of the control, in PowerBuilder units.
FillColor	Long	Specifies the numeric value of the color used to fill the control: -2 to 16,777,215. For more information about color, see RGB in the *PowerScript Reference*.
FillPattern	FillPattern (enumerated)	Specifies the hatch pattern used to fill the control. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower right.
Height	Integer	Specifies the height of the control, in PowerBuilder units.

RoundRectangle property	Datatype	Description
LineColor	Long	Specifies the numeric value of the line color: -2 to 16,777,215. For more information about color, see RGB in the *PowerScript Reference*.
LineStyle	LineStyle (enumerated)	Specifies the style of the line used to draw the control. Values are: Continuous! Dash! DashDot! DashDotDot! Dot! Transparent!
LineThickness	Integer	Specifies the thickness of the line used to draw the control, in PowerBuilder units. If LineThickness is greater than one pixel (about four PowerBuilder units), the LineStyle is Continuous!.
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

RoundRectangle event	Occurs
Constructor	Immediately before the Open event occurs in the window.
Destructor	Immediately after the Close event occurs in the window.

Functions

RoundRectangle function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.

RoundRectangle function	Datatype returned	Description
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
Resize	Integer	Changes the size of the control.
Show	Integer	Makes the control visible.
TypeOf	Object	Returns the type of the control.

RuntimeError object

The RuntimeError object inherits from the Throwable object and is used by the PowerBuilder virtual machine (PBVM) to throw runtime errors. Runtime errors are also called unchecked exceptions. You do not need to declare where they might be thrown and you do not need to catch them as you do checked exceptions.

When a RuntimeError is thrown, its properties are populated automatically with the runtime information associated with the line where the error occurred. If a RuntimeError is not handled, the Application object SystemError event is triggered and the global Error object is populated with the runtime information.

The following derived types provide more robust error-handling capabilities:

- DivideByZeroError – thrown when an attempt is made to divide by zero.
- NullObjectError – thrown when an attempt is made to access an object using a null reference.
- PBXRuntimeError – thrown when an unknown error occurs in a PowerBuilder extension.
- CORBASystemException – thrown when a CORBA system exception is thrown from EAServer.
- DWRuntimeError – thrown when a DataWindow error occurs that is not handled by an Error event script.
- OLERuntimeError – thrown when an OLE error occurs that is not handled by an ocx_error, ExternalException, or Error event script

Additional objects that map to standard CORBA exception types inherit from CORBASystemException. The PowerBuilder exception class name is the same as the CORBA exception name without underscore characters. For example, CORBAFreeMem maps to CORBA_FREE_MEM. You can view the list of CORBASystemException types in the PowerBuilder System Tree or in the Browser.

The descendants of RuntimeError allow you to handle specific runtime errors. For example, you can catch only NullObjectError exceptions in a specific block of code. Alternatively, you can catch all runtime errors with a single CATCH statement. Except for PBXRuntimeError, the error information available in the descendant objects is also available in the RuntimeError object.

PBXRuntimeError has an additional property, DLLName, that identifies the name of the PowerBuilder extension DLL in which the error occurred.

Properties

RuntimeError property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Class	String	Name of the class where the exception occurred
Description	String	(OLERuntimeError only) Textual description of the exception
DLLName	String	(PBXRuntimeError only) Name of the PowerBuilder extension DLL where the exception occurred
HelpFile	String	(OLERuntimeError only) Full file name of Help file containing information about the exception
HelpContext	UnsignedLong	(OLERuntimeError only) Help context ID of the topic in the Help file containing information about the exception
Line	Integer	Line number where the exception occurred
Number	Integer	Number of the PowerBuilder error
ObjectName	String	Name of the object where the exception occurred
RoutineName	String	Name of the event or routine where the exception occurred
Source	String	(OLERuntimeError only) Source of the exception
Text	String	Text associated with the type of exception

Events

RuntimeError event	Occurs
Constructor	When the exception is thrown
Destructor	Immediately after the exception is thrown

Functions

RuntimeError function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetMessage	String	Returns the error message from objects of type Throwable
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
SetMessage	—	Sets an error message for an object of type Throwable
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

ScriptDefinition object

Information about a script associated with a class definition. ScriptDefinition is used in the ClassDefinition object.

You cannot instantiate a ScriptDefinition object for a particular script independently of a ClassDefinition object. Instead you access the ScriptDefinition instances that are elements of the ScriptList array of a ClassDefinition instance.

The ScriptDefinition object has information about:

- The script's name and whether it is a function or an event
- The return type, arguments, and local variables
- The source code
- Whether the script is defined locally or in an ancestor

- External function declarations

A ScriptDefinition object has no events.

Properties

ScriptDefinition property	Datatype	Description
Access	VarAccess	The access level of the script (what objects can call the script). Values are: Private! Public! Protected! System!
AliasName	String	The alias value for an external function. The value is an empty string ("") for scripts that are not aliased external functions. Corresponds to the ALIAS FOR keyword in the external function declaration.
ArgumentList	VariableDefinition	An unbounded array whose elements are VariableDefinition objects, one object per argument. The array is empty if there are no arguments.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
EventId	Long	The numeric event ID for an event. For events that do not have an ID, the value is -1.
EventIdName	String	The event ID name for an event. For events that do not have an ID, the value is an empty string ("").
ExternalUser Function	String	The file name of the DLL containing the external user function. The value is an empty string ("") for scripts that are not external user functions. Corresponds to the LIBRARY keyword in the external function declaration.
IsExternalEvent	Boolean	Indicates whether this is an external event. External events are automatically generated events that get dispatched elsewhere.
IsLocallyDefined	Boolean	Indicates whether the event is defined at this level in the inheritance hierarchy. Values are: TRUE – The event is defined at this level of the object's inheritance hierarchy. FALSE – The event is defined at an ancestor level. IsLocallyDefined is not applicable to functions.

ScriptDefinition property	Datatype	Description
IsLocallyScripted	Boolean	Indicates whether the script is implemented at this level in the inheritance hierarchy. Values are: TRUE – There is code for the event or function at this level of the object's inheritance hierarchy. FALSE – There is no code for the event or function at this level.
IsRPCFunction	Boolean	Whether this is an RPC function. Values are: TRUE – Is an RPC function. FALSE – Is not an RPC function. Corresponds to the RPCFUNC keyword in a declaration for a stored procedure.
IsScripted	Boolean	Whether the event has a definition but no code at any level of the collapsed inheritance hierarchy. Values are: TRUE – The event has a script at some level of the object's inheritance hierarchy. FALSE – The event does not have a script. Only events can be defined but not scripted. For functions, IsScripted is always true.
Kind	ScriptKind	Whether the script is a function or event. Values are: ScriptEvent! ScriptFunction!
LocalVariableList	VariableDefinition	An unbounded array whose elements are VariableDefinition objects, one object per local variable. The array is empty if there are no local variables.
Name	String	The name of the script.
ReturnType	TypeDefinition	The type information of the return value. For scripts that do not return anything, ReturnType is an invalid object. Use the IsValid function to test the value.
Source	String	The source code for the script. Source is an empty string ("") if the source is not available (for example, when running an executable).
SystemFunction	String	For built-in PowerBuilder functions, the file name of the DLL containing the function. The value is an empty string ("") for scripts that are not built-in PowerBuilder functions.

Functions

ScriptDefinition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

SimpleTypeDefinition object

Information about the type of a scalar variable. SimpleTypeDefinition is inherited from TypeDefinition and has no additional properties or functions.

For the list of properties and functions, see the TypeDefinition object.

SingleLineEdit control

A SingleLineEdit is a box in which the user can enter a single line of text. You typically use a SingleLineEdit as an input field.

Properties

SingleLineEdit property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the key you want to assign as the accelerator key for a control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.

SingleLineEdit property	Datatype	Description
AutoHScroll	Boolean	Specifies whether the control automatically scrolls horizontally when data is entered or deleted. Values are: TRUE – Control automatically scrolls horizontally. FALSE – Control does not automatically scroll horizontally.
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see RGB in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisplayOnly	Boolean	Specifies whether the text in the control is display only and cannot be changed by the user. Values are: TRUE – Text cannot be changed by user. FALSE – Text can be changed by user.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.

SingleLineEdit property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HideSelection	Boolean	Specifies whether selected text stays selected (highlighted) even when the control does not have focus. Values are: TRUE – Text does not stay highlighted. FALSE – Text stays highlighted.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Limit	Integer	Specifies the maximum number of characters (0 to 32,767) that can be entered in the control (0 means unlimited).
Password	Boolean	Specifies whether the control is a password field (whether asterisks appear when the user types characters). Values are: TRUE – Control is a password field. FALSE – Control is not a password field.

SingleLineEdit property	Datatype	Description
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextCase	TextCase (enumerated)	Specifies the case in which text entered in the control displays. Values are: AnyCase! Lower! Upper!
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

SingleLineEdit event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Modified	When the control text has been changed and the user presses Enter or Tab or changes focus to another control
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

SingleLineEdit function	Datatype returned	Description
CanUndo	Boolean	Returns true if the Undo function can be used to undo the last edit in the control and returns false if it cannot.
ClassName	String	Returns the name assigned to the control.
Clear	Integer	Clears the selected text (if any) from the control (but does not place it in the clipboard).
Copy	Integer	Copies (but does not delete) the selected text (if any) from the control to the clipboard.
Cut	Integer	Cuts (deletes) the selected text (if any) from the control and places it in the clipboard.
Drag	Integer	Starts or ends the dragging of a control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.

SingleLineEdit function	Datatype returned	Description
Paste	Integer	Inserts the contents of the clipboard (if any) at the insertion point in the control and replaces the selected text (if any).
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
Position	Integer	Returns the position of the insertion point in the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
ReplaceText	Integer	Replaces the currently selected text (if any) with the specified string. If no text is selected, inserts the text at the current insertion point.
Resize	Integer	Changes the size of the control.
SelectedLength	Integer	Returns the length of the selected text (if any) in the control.
SelectedStart	Integer	Returns the starting position of the selected text (if any) in the control.
SelectedText	String	Returns a string with the selected text (if any) from the control.
SelectText	Integer	Selects the text in the control specified by the starting position and length.
SetFocus	Integer	Sets focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.
Undo	Integer	Cancels the previous editing function performed in the control.

SSLCallBack object (obsolete)

Obsolete object
SSLCallBack object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The SSLCallBack object provides PowerBuilder clients with the ability to handle SSL callbacks from EAServer.

Properties

SSLCallBack property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Events

SSLCallBack event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

SSLCallBack function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetCertificateLabel (obsolete)	String	Allows an SSL client to select a certificate label to use from a list of certificate labels passed as an argument to the function
GetCredentialAttribute (obsolete)	String	Allows an SSL client to supply user credentials dynamically
GetParent	PowerObject	Returns a reference to the name of the parent object
GetPin (obsolete)	String	Obtains a PIN for use with an SSL connection
PostEvent	Boolean	Adds an event to the end of the message queue for the object

SSLCallBack function	Datatype returned	Description
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TrustVerify (obsolete)	Long	Allows an SSL client to approve a certificate chain for use
TypeOf	Object	Returns the type of the object

SSLServiceProvider object (obsolete)

Obsolete object
SSLServiceProvider object is obsolete, because EAServer is no longer supported since PowerBuilder 2017.

The SSLServiceProvider object allows you to establish a Secure Sockets Layer (SSL) connection from a PowerBuilder client to EAServer.

Properties

SSLServiceProvider property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

SSLServiceProvider event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

SSLServiceProvider function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetGlobalProperty (obsolete)	String	Returns the value of an SSL global property
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
SetGlobalProperty (obsolete)	Long	Sets the value of an SSL global property
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event
TypeOf	Object	Returns the type of the object

StaticHyperLink control

The StaticHyperLink control is a descendant of the StaticText control. The URL property of the StaticHyperLink control enables you to provide a hot link to a Web page. When the user clicks the control, the user's Web browser opens to display the page you specify.

Usage note
If you know that your users have browsers that support URL completion, you can enter a partial address, such as `appeon.com`.You can always enter a complete address, such as `http://www.appeon.com`.

Properties

StaticHyperLink property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.

StaticHyperLink property	Datatype	Description
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the text alignment in the control. Values are: Left! Center! Right!
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderColor	Long	Specifies the numerical value of the border color: -2 to 16,777,215.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisabledLook	Boolean	Specifies whether the control appears to be enabled.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.

StaticHyperLink property	Datatype	Description
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Courier).
FillPattern	FillPattern (enumerated)	Specifies the hatch pattern used to fill the control. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower right.
FocusRectangle	Boolean	Specifies whether a dotted rectangle (focus rectangle) frames the control when it has focus. Values are: TRUE – Control framed when it has focus FALSE – Control not framed when it has focus
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!

StaticHyperLink property	Datatype	Description
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the rectangular box that contains the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 is the default and means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextColor	Long	Specifies the numeric value of the text color in the control: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
URL	String	Specifies the URL to open in the user's Web browser when the text control is clicked, provided no Clicked event is coded. The status text displays the URL when the mouse passes over the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.

StaticHyperLink property	Datatype	Description
Width	Integer	Specifies the width of the rectangular box that contains the control, in pixels.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

StaticHyperLink event	Occurs
Clicked	When the control is clicked (selected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

StaticHyperLink function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible

StaticHyperLink function	Datatype returned	Description
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the rectangular box that contains the control
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

StaticText control

StaticText is display text that the user can select but cannot modify with the keyboard. You can explicitly modify the StaticText in a script.

Properties

StaticText property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the text alignment in the control. Values are: Left! Center! Right!

StaticText property	Datatype	Description
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderColor	Long	Specifies the numerical value of the border color: -2 to 16,777,215.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DisabledLook	Boolean	Specifies whether the control appears to be enabled.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Control can be selected. FALSE – Control cannot be selected.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, Arial or Courier).

StaticText property	Datatype	Description
FillPattern	FillPattern (enumerated)	Specifies the hatch pattern used to fill the control. Values are: BDiagonal! Diamond! FDiagonal! Horizontal! Solid! Square! Vertical! FDiagonal! is lines going from the lower-left to the upper-right. BDiagonal! is lines going from the upper-left to the lower right.
FocusRectangle	Boolean	Specifies whether a dotted rectangle (focus rectangle) frames the control when it has focus. Values are: TRUE – Control framed when it has focus. FALSE – Control not framed when it has focus.
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font used for the text in the control. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the rectangular box that contains the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.

StaticText property	Datatype	Description
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order.
TabOrder	Integer	Specifies the tab value of the control within the window (0 is the default and means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Text	String	Specifies the text that displays in the control.
TextColor	Long	Specifies the numeric value of the text color in the control: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the rectangular box that contains the control, in pixels.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

StaticText event	Occurs
Clicked	When the control is clicked (selected)
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the control is double-clicked (selected and activated)

StaticText event	Occurs
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

StaticText function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the rectangular box that contains the control.
SetFocus	Integer	Sets focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

SyncParm object

A SyncParm object is a system structure that you can use to obtain runtime properties from a synchronization option window, and then pass these property values to an MLSync object. The SyncParm object has no events or functions.

Properties

SyncParm property	Datatype	Description
AuthenticateParms	String	Used to pass parameters to an authenticate_parameters connection script.
DBPass	String	Password for the SQL Anywhere remote database.
DBUser	String	User ID for the SQL Anywhere remote database.
EncryptionKey	String	Encryption key for SQL Anywhere remote database.
MLPass	String	The MobiLink password passed to the synchronization server.
MLUser	String	The MobiLink user name passed to the synchronization server.
ReturnCode	Long	Return code from the synchronization options window (generated by the MobiLink synchronization wizard). Values are: • 0 = Success. • -1 = Error. • 100 = Cancel.
UlTrans	Transaction	Not currently used. Reserved for the connected transaction object to an UltraLite remote database.

Tab control

A Tab control contains tab pages, which are user objects that contain controls. Tab pages can be defined within the Tab control or they can be defined in the User Object painter and inserted into the Tab control.

Each tab page can have its own label, picture, and background color.

All tab pages share the same font settings.

Tab Position controls where the tabs on the tab pages are displayed. Tabs can be displayed on any one of the four edges of the Tab control. They can also be displayed on opposite edges with the tabs before or after the selected tab jumping to the other edge.

Properties

Tab property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
Alignment	Alignment (enumerated)	Specifies the alignment of the text labeling all the tabs. Values are: Left! Center! Right!
BackColor	Long	Specifies the numeric value of the background color: –2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
BoldSelectedText	Boolean	Specifies whether the text for the selected tab is bold. Values are: TRUE – The text on the selected tab is bold. FALSE – The text on the selected tab has the same setting as the other tabs.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order in the window. Values are: TRUE – Moves to the top. FALSE – Does not move to the top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Tab property	Datatype	Description
CreateOnDemand	Boolean	Specifies whether PowerBuilder creates graphical representations of controls on all tab pages when the Tab control is created. Values are: TRUE – Graphical representations of tab pages are not created until the tab page is selected. FALSE – (Default) Graphical representations of tab pages are created when the Tab control is created.
Control[ ]	UserObject	Specifies the array of tab pages within the Tab control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the control is enabled (can be selected). Values are: TRUE – Can be selected. FALSE – Cannot be selected.
FaceName	String	Specifies the name of the typeface used for the text labels on tabs (for example, Arial or Courier). Only TrueType fonts display correctly on vertical tabs.
FixedWidth	Boolean	Specifies whether tabs have a fixed width, meaning they do not shrink to the length of their text labels. Values are: TRUE – Tab width is fixed; the width is determined by the longest text label. FALSE – Tab width adjusts to the length of the text labels.

Tab property	Datatype	Description
FocusOnButtonDown	Boolean	Specifies whether each tab gets focus when the user clicks on it. Values are: TRUE – The tab the user clicks on gets focus; a dotted rectangle marks the tab. FALSE – The tab does not get focus. In either case, the selected tab page comes to the front. The dotted focus rectangle appears on the tab when the user clicks on it a second and subsequent times even if this property is set to false.
FontCharSet	FontCharSet (enumerated)	Specifies the character set for the text labels on the tabs. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) for the text labels on the tabs. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the pitch (spacing) of the font for text labels on the tabs. Values are: Default! Fixed! Variable!
Height	Integer	Specifies the height of the control, in PowerBuilder units.
Italic	Boolean	Specifies whether the text on the tabs is italic. Values are: TRUE – Text labels are italic. FALSE – Text labels are not italic.
Multiline	Boolean	Specifies whether the tabs can appear in more than one row. Values are: TRUE – If there is not room for all the tabs in a single row, the tabs are arranged in multiple rows. FALSE – If there is not room for all the tabs in a single row, a dual arrow control displays to allow the user to scroll to tabs that do not fit.

Tab property	Datatype	Description
PerpendicularText	Boolean	Specifies whether the tab labels are drawn perpendicular to the tab page. Values are: TRUE – Text is perpendicular to the edge of the tab page, resulting in narrower tabs. FALSE – Text runs along the edge of the tab page, resulting in wider tabs.
PictureOnRight	Boolean	Specifies whether a picture that is part of the tab label is to the right or left of the text. Values are: TRUE – The picture is on the right. FALSE – The picture is on the left.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
PowerTips	Boolean	Specifies whether PowerTipText for a tab page is displayed as a PowerTip (a pop-up label for the tab) when the mouse pointer pauses over the tab. PowerTips are useful if the tabs are labeled with pictures. Values are: TRUE – PowerTipText, if any, displays as a pop-up label for each tab. FALSE – No PowerTips are displayed.
RaggedRight	Boolean	Specifies whether tabs are stretched so that they fill space along the edge of the control. Values are: TRUE – Tabs remain the size determined by their label text and the FixedWidth property. FALSE – Tabs are stretched to fill the edge.
SelectedTab	Integer	Specifies the index number of the selected tab. The default value is 1, and the integer must be in the range 1 to N, where N is the number of tab pages.
ShowPicture	Boolean	Specifies whether the picture selected for each tab is displayed. Values are: TRUE – The picture for each tab, if any, is displayed. FALSE – No pictures are displayed.
ShowText	Boolean	Specifies whether the text specified for each tab label is displayed. Values are: TRUE – The text for each tab is displayed. FALSE – The text for each tab is not displayed.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).

Tab property	Datatype	Description
TabPosition	TabPosition (enumerated)	Specifies where the tabs appear around the Tab control. Values are: • TabsOnBottom! – Tabs are at the bottom. • TabsOnBottomAndTop! – Tabs before the selected tab are on top; the selected tab itself and tabs after it are on the bottom. • TabsOnLeft! – Tabs are on the left. • TabsOnLeftAndRight! – Tabs before the selected tab and the selected tab itself are on the left; tabs after the selected tab are on the right. • TabsOnRight! – Tabs are on the right. • TabsOnRightAndLeft! – Tabs before the selected tab are on the left; the selected tab and tabs after it are on the right. • TabsOnTop! – Tabs are on top. • TabsOnTopAndBottom! – Tabs before the selected tab and the selected tab itself are on top; tabs after the selected tab are on the bottom.
Tag	String	Specifies the tag value assigned to the control.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
Underline	Boolean	Specifies whether the text on the tabs is underlined. Values are: TRUE – Text labels are underlined. FALSE – Text labels are not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Is visible. FALSE – Is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

Tab Event	Occurs
Clicked	When the user clicks in the Tab control, except in the display area of the tab page. For the tab page, the Clicked event (pbm_bnclicked) for the user object is triggered instead.
Constructor	When the object is created, immediately before the Open event occurs in the window.
Destructor	When the object is destroyed, immediately after the Close event occurs in the window.
DoubleClicked	When the user double-clicks in the Tab control, except in the display area of the tab page. For the tab page, the DoubleClicked event (pbm_bndoubleclicked) for the user object is triggered instead.
DragDrop	When a dragged control is dropped in the tab area of the control.
DragEnter	When a dragged control enters the control, including entering the narrow border around the display area.
DragLeave	When a dragged control leaves the control, including leaving by crossing into the tab page display area.
DragWithin	When a dragged control is within the control. but not within the tab page display area.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
Key	When the user presses a key.
LoseFocus	When the control loses focus (becomes inactive).
Other	When a Windows message occurs that is not a PowerBuilder event.
RightClicked	When the user clicks with the right mouse button in the Tab control, except in the display area of the tab page. If the user right-clicks on the tab page, the controls or user objects on the tab page get an RButtonDown event.
RightDoubleClicked	When the user double-clicks with the right mouse button in the Tab control, except in the display area of the tab page. For the tab page, the RightDoubleClicked event (pbm_rbuttondblclk) for the user object is triggered instead.
SelectionChanged	Just after the selection changes to another tab. SelectionChanged is triggered when the tab is created and the initial selection is established.
SelectionChanging	Just before the selection changes to another tab. To prevent the selection from changing, return 1 in the event script. SelectionChanging is triggered when the tab is created and the initial selection is established.

Functions

Tab function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
CloseTab	Integer	Closes a tab page that was opened with the OpenTab function.
Drag	Integer	Starts or ends dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the parent of the Tab control.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
MoveTab	Integer	Moves a tab to a new position in the order of tabs.
OpenTab	Integer	Opens the specified user object as a tab page, making its properties available to scripts.
OpenTabWithParm	Integer	Opens the user object as a tab page, making its properties available to scripts, and stores a parameter in the Message object.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Integer	Adds the specified event to the end of the event queue for the specified object.
Print	Integer	Prints the contents of the control. You can specify a range of pages and other settings.
Resize	Integer	Changes the size of the tab page to the size specified in the width and height arguments.
SelectTab	Integer	Selects a tab page.
SetFocus	Integer	Sets focus to the Tab control.
SetPosition	Integer	Specifies the control's position in the front-to-back order within the window.
SetRedraw	Integer	Controls automatic redrawing of the control.
Show	Integer	Makes the control visible.
TabPostEvent	Integer	Adds an event to the end of the message queues for each of the tab pages.
TabTriggerEvent	Integer	Sends an event to every tab page and, for each page, executes the script associated with the event.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

Throwable object

The Throwable datatype is the base class for all throwable objects. These include exceptions and error objects.

Properties

Throwable property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Text	String	Contains the text of the error message.

Events

Throwable event	Occurs
Constructor	When the exception is thrown.
Destructor	Immediately after the exception is thrown.

Functions

Throwable function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetMessage	String	Returns the error message from objects of type Throwable.
GetParent	PowerObject	Returns a reference to the name of the parent object.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
SetMessage	—	Sets an error message for an object of type Throwable.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

Timing object

Timing is a nonvisual system object that can be used when a Timer event cannot be associated with a window. To use a timing object, create a standard class user object that inherits from the Timing system class, and then create an instance of the inherited timing object.

Properties

Timing property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Interval	Double	(Read-only) The minimum number of seconds the timing object waits between calls to the Timer event. This property is initially set to NULL and is modified whenever the Start function is called.
Running	Boolean	(Read-only) True if the timing object has been started and is currently running. False if the timing object is not running.

Events

Timing event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
Timer	When a specified number of seconds elapses after the Start function has been called

Functions

Timing function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for the object
Start	Integer	Activates the timing object using a specified interval
Stop	Integer	Deactivates the timing object
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event

Timing function	Datatype returned	Description
TypeOf	Object	Returns the type of the object

TraceActivityNode object

The TraceActivityNode object provides information about the nodes in a trace file, including the type of activity represented by a node. You use the TraceActivityNode object in conjunction with the TraceFile object.

The TraceActivityNode object has no events.

Properties

TraceActivityNode property	Datatype	Description
ActivityType	TraceActivity (enumerated)	A value of the enumerated datatype TraceActivity that identifies the activity represented by the node. Values are: ActBegin! – Start and finish of logging. ActError! – Occurrences of system errors and warnings. ActESQL! – Embedded SQL statement entry and exit. ActGarbageCollect! – Start and finish of garbage collection ActLine! – Routine line hits. ActObjectCreate! – Object creation. ActObjectDestroy! – Object destruction. ActRoutine! – Routine entry and exit. ActUser! – Occurrences of an activity you selected.
Category	TraceCategory (enumerated)	The category of the activity represented by the node. Values are: TraceAtomic! – The node is an activity that occurred in a single statement. TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceActivityNode function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceBeginEnd object

The TraceBeginEnd object provides information about a node in a trace file identified as an occurrence of a logging start or finish. To access the extra properties of the TraceBeginEnd object, you assign a TraceActivityNode object whose activity type is ActBegin! to the TraceBeginEnd object.

The TraceBeginEnd object has no events.

Properties

TraceBeginEnd property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActBegin! which identifies the activity represented by the node as an occurrence of a logging start or finish.
Category	TraceCategory (enumerated)	The category of the activity represented by the node. Values are: TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Message	String	The message passed in for the TraceBegin function. For the TraceEnd function, the value is " ".
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceBegin End function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceError object

The TraceError object provides information about a node in a trace file identified as an occurrence of a system error or warning, including the error message and severity level. To access the extra properties of the TraceError object, you assign a TraceActivityNode object whose activity type is ActError! to the TraceError object.

The TraceError object has no events.

Properties

TraceError property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActError! which identifies the activity represented by the node as an occurrence of a system error or warning
Category	TraceCategory (enumerated)	The value TraceAtomic! which indicates that the node is an activity that occurred in a single statement
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Message	String	The system error message or the message passed to the TraceError function
Severity	Long	The system error severity or the severity argument passed to the TraceError function
TimerValue	Decimal	The timer value (in seconds) when the activity occurred

Functions

TraceError function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceESQL object

The TraceESQL object provides information about a node in a trace file identified as an occurrence of an Embedded SQL (ESQL) statement. To access the extra properties of the TraceESQL object, you assign a TraceActivityNode object whose activity type is ActESQL! to the TraceESQL object.

The TraceESQL object has no events.

Properties

TraceESQL property	Datatype	Description
ActivityNode	TraceActivity	The value ActESQL! which identifies the activity represented by the node as an occurrence of an ESQL statement entry or exit.
Category	TraceCategory	The category of the activity represented by the node. Values are: TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Name	String	The name of the ESQL statement.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceESQL function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceFile object

The TraceFile object is used to access the contents of a trace file created from a PowerBuilder application. Unlike the Profiling and TraceTree objects, the TraceFile object does not provide properties and functions to create an analysis model. You use the TraceFile object in conjunction with the TraceActivityNode, TraceBeginEnd, TraceError, TraceESQL, TraceGarbageCollect, TraceLine, TraceObject, TraceRoutine, and TraceUser objects.

The TraceFile object has no events.

Properties

TraceFile property	Datatype	Description
ApplicationName	String	The name of the application used to generate the trace file.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CollectionTime	Decimal	The amount of time (in seconds) taken by the collection of trace data. This time has already been accounted for in the timestamps from the trace file, that is, the proper amount of time has been subtracted from the timestamps before they are put in the trace file. The value is NULL if the file is not open.
LastError	ErrorReturn (enumerated)	The error code for the last error that occurred.

TraceFile property	Datatype	Description
NumberOfActivities	Long	The number of activities that exist in the trace file.
FileName	String	The name of the opened trace file. The value is an empty string if the file is not open.

Functions

TraceFile function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
Close	ErrorReturn (enumerated)	Closes the open trace file.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
NextActivity	TraceActivityNode	Provides the next activity in the trace stream. If there are no more activities, or if the file is not open, an invalid object is returned. Use the LastError property to determine what kind of error occurred.
Open	ErrorReturn (enumerated)	Opens for reading the trace file with the passed name.
Reset	ErrorReturn (enumerated)	Resets the stream to the beginning of the trace file.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceGarbageCollect object

The TraceGarbageCollect object provides information about a node in a trace file identified as an occurrence of garbage collection. To access the extra properties of the TraceGarbageCollect object, you assign a TraceActivityNode object whose activity type is ActGarbageCollect! to the TraceGarbageCollect object.

The TraceGarbageCollect object has no events.

Properties

TraceGarbageCollect property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActGarbageCollect! which identifies the activity represented by the node as an occurrence of garbage collection start or finish.
Category	TraceCategory (enumerated)	The category of the activity represented by the node. Values are: TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceGarbageCollect function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceLine object

The TraceLine object provides information about a node in a trace file identified as an occurrence of a routine line hit. To access the extra properties of the TraceLine object, you assign a TraceActivityNode object whose activity type is ActLine! to the TraceLine object.

The TraceLine object has no events.

Properties

TraceLine property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActLine! which identifies the activity represented by the node as an occurrence of a routine line hit
Category	TraceCategory (enumerated)	The value TraceAtomic! which indicates that the node is an activity that occurred in a single statement
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
LineNumber	UnsignedLong	The line number
TimerValue	Decimal	The timer value (in seconds) when the activity occurred

Functions

TraceLine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceObject object

The TraceObject object provides information about a node in a trace file identified as the creation or destruction of an object. To access the extra properties of the TraceObject object, you assign a TraceActivityNode object whose activity type is ActObjectCreate! or ActObjectDestroy! to the TraceObject object. The TraceObject object has no events.

Properties

TraceObject property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActObjectCreate! or ActObjectDestroy! which identifies the activity represented by the node as the creation or destruction of an object.
Category	TraceCategory (enumerated)	The category of the activity represented by the node. Values are: TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassName	String	The name of the class that is the type of the object. Nested classes (like controls in windows) have a name of the form *class name`embedded class name*.
IsCreate	Boolean	True if the node represents the creation of an object and false if the node represents the destruction of an object.
LibraryName	String	The name of the library that contains the class of the object. The value is " " for system classes.
ObjectID	UnsignedLong	A unique identifier for the object.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceObject function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceRoutine object

The TraceRoutine object provides information about a node in a trace file identified as an occurrence of a routine. To access the extra properties of the TraceRoutine object, you assign a TraceActivityNode object whose activity type is ActRoutine! to the TraceRoutine object. The TraceRoutine object has no events.

Properties

TraceRoutine property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActRoutine! which identifies the activity represented by the node as an occurrence of a routine entry or exit.
Category	TraceCategory (enumerated)	The category of the activity represented by the node. Values are: TraceIn! – The node is the beginning of an activity that spans several statements. TraceOut! – The node is the end of an activity that spanned several statements.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassName	String	The name of the class that contains the routine. The value is " " for system functions. Nested classes (like controls in windows) have a name of the form *class name`embedded class name*.
IsEvent	Boolean	True if the routine is an event and false if the routine is a function.
LibraryName	String	The name of the library that contains the class that includes the routine. The value is " " for system classes.
Name	String	The name of the routine including the parameter datatypes and return value.
ObjectID	UnsignedLong	The object ID for the object on which the routine is executing. The ID is 0 if the routine executing is a global or system routine.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceRoutine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service

TraceRoutine function	Datatype returned	Description
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceTree object

The TraceTree object is used to analyze the performance of a PowerBuilder application. It provides a tree model listing all the nodes logged in a given trace file. It includes the functions you call to name the trace file to be analyzed, build the tree model, and list the top-level entries in the tree model. You use the TraceTree object in conjunction with the TraceTreeNode, TraceTreeError, TraceTreeESQL, TraceTreeGarbageCollect, TraceTreeLine, TraceTreeObject, TraceTreeRoutine, and TraceTreeUser objects.

The TraceTree object has no events.

Properties

TraceTree property	Datatype	Description
ApplicationName	String	The name of the application used to generate the trace file.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CollectionTime	Decimal	The amount of time (in seconds) taken by the collection of trace data. This time has already been accounted for in the timestamps from the trace file (the proper amount of time has been subtracted from the timestamps before they are put in the trace file). The value is NULL if no model was built.
NumberOfActivities	Long	The total number of activities that exist in the trace file. The value is 0 if this is called before the trace file name is set.
TraceFileName	String	The name of the trace file to use to build the model. The value is an empty string if the name has not been successfully set.

Functions

TraceTree function	Datatype returned	Description
BuildModel	ErrorReturn (enumerated)	Builds a tree model based on the previously specified trace file
ClassName	String	Returns the name assigned to the object
DestroyModel	ErrorReturn (enumerated)	Destroys the current tree model
EntryList	ErrorReturn (enumerated)	Provides a list of top-level entries (defined as TraceTreeNode objects) included in the model
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
SetTraceFileName	ErorReturn	Indicates the name of the trace file to use for analysis and creates the format of the file header
TypeOf	Object (enumerated)	Returns the type of the object

TraceTreeError object

The TraceTreeError object provides information about a tree model node identified as an occurrence of a system error or warning, including the error message and severity level. To access the extra properties of the TraceTreeError object, you assign a TraceTreeNode object whose activity type is ActError! to the TraceTreeError object.

The TraceTreeError object has no events.

Properties

TraceTreeError property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActError! which identifies the activity represented by the node as an occurrence of a system error or warning
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

TraceTreeError property	Datatype	Description
Message	String	The system error message or the message passed to the TraceError function
ParentNode	TraceTreeNode	The parent of this node
Severity	Long	The system error severity or the severity argument passed to the TraceError function
TimerValue	Decimal	The timer value (in seconds) when the activity occurred

Functions

TraceTree-Error function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceTreeESQL object

The TraceTreeESQL object provides information about a tree model node identified as an occurrence of an Embedded SQL (ESQL) statement. To access the extra properties of the TraceTreeESQL object, you assign a TraceTreeNode object whose activity type is ActESQL! to the TraceTreeESQL object.

The TraceTreeESQL object has no events.

Properties

TraceTreeESQL property	Datatype	Description
ActivityNode	TraceActivity (enumerated)	The value ActESQL! which identifies the activity represented by the node as an occurrence of an ESQL statement entry and exit

TraceTreeESQL property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
EnterTimerValue	Decimal	The timer value (in seconds) of the entry for this statement
ExitTimerValue	Decimal	The timer value (in seconds) of the exit for this statement
Name	String	The name of the ESQL statement
ParentNode	TraceTreeNode	The parent of this node

Functions

TraceTreeESQL function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceTreeGarbageCollect object

The TraceTreeGarbageCollect object provides information about a tree model node identified as an occurrence of garbage collection, including the children or classes and routines called by that node. To access the extra properties of the TraceTreeGarbageCollect object, you assign a TraceTreeNode object whose activity type is ActGarbageCollect! to the TraceTreeGarbageCollect object.

The TraceTreeGarbageCollect object has no events.

Properties

TraceTreeGarbageCollect property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActGarbageCollect! which identifies the activity represented by the node as garbage collection start and finish.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
EnterTimerValue	Decimal	The timer value (in seconds) of the entry for the garbage collector.
ExitTimerValue	Decimal	The timer value (in seconds) of the exit for the garbage collector.
ParentNode	TraceTreeNode	The parent of this node.

Functions

TraceTreeGarbageCollect function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetChildrenList	ErrorReturn (enumerated)	Provides a list of the children (defined as TraceTreeNode objects) of this routine
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceTreeLine object

The TraceTreeLine object provides information about a tree model node identified as an occurrence of a routine line hit. To access the extra properties of the TraceTreeLine object, you assign a TraceTreeNode object whose activity type is ActLine! to the TraceTreeLine object.

The TraceTreeLine object has no events.

Properties

TraceTreeLine property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActLine! which identifies the activity represented by the node as an occurrence of a routine line hit
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
LineNumber	UnsignedLong	The line number
ParentNode	TraceTreeNode	The parent of this node
TimerValue	Decimal	The timer value (in seconds) when the activity occurred

Functions

TraceTreeLine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceTreeNode object

The TraceTreeNode object provides information about the nodes in the tree model, including the type of activity represented by the node. You use the TraceTreeNode object in conjunction with the TraceTree object.

The TraceTreeNode object has no events.

Properties

TraceTreeNode property	Datatype	Description
ActivityType	TraceActivity (enumerated)	A value of the enumerated datatype TraceActivity that identifies the activity represented by the node. Values are: ActBegin! – Start and finish of logging. ActError! – Occurrences of system errors and warnings. ActESQL! – Embedded SQL statement entry and exit. ActGarbageCollect! – Start and finish of garbage collection. ActLine! – Routine line hits. ActObjectCreate! – Object creation. ActObjectDestroy! – Object destruction. ActRoutine! – Routine entry and exit. ActUser! – Occurrences of an activity you selected.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ParentNode	TraceTreeNode	The parent of this node. If the parent is a top-level node, that is, a node returned by the EntryList function, the value is an invalid object.

Functions

TraceTreeNode function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceTreeObject object

The TraceTreeObject object provides information about a tree model node identified as an occurrence of an object. To access the extra properties of the TraceTreeObject object, you assign a TraceTreeNode object whose activity type is ActObjectCreate! or ActObjectDestroy! to the TraceTreeObject object.

The TraceTreeObject object has no events.

Properties

TraceTreeObject property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActObjectCreate! or ActObjectDestroy! which identifies the activity represented by the node as object creation or destruction.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassName	String	The name of the class that is the object type.
EnterTimerValue	Decimal	The timer value (in seconds) when the activity began.
ExitTimerValue	Decimal	The timer value (in seconds) when the activity ended.
IsCreate	Boolean	True if the node represents the creation of an object and false if it represents the destruction of an object.
ObjectID	UnsignedLong	The internal identifier for the object.
ParentNode	TraceTreeNode	The parent of this node.

Functions

TraceTreeObject function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetChildrenList	ErrorReturn (enumerated)	Provides a list of the children (defined as TraceTreeNode objects) of this object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceTreeRoutine object

The TraceTreeRoutine object provides information about a tree model node identified as an occurrence of a routine. To access the extra properties of the TraceTreeRoutine object, you assign a TraceTreeNode object whose activity type is ActRoutine! to the TraceTreeRoutine object.

The TraceTreeRoutine object has no events.

Properties

TraceTreeRoutine property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActRoutine! which identifies the activity represented by the node as an occurrence of a routine entry and exit.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClassName	String	The name of the class that contains the routine. The value is " " for system functions. Nested classes (like controls on a window) have a name of the form *class name`embedded class name*.
EnterTimerValue	Decimal	The timer value (in seconds) of the entry for this call.
ExitTimerValue	Decimal	The timer value (in seconds) of the exit for this call.
IsEvent	Boolean	True if the routine is an event and false if the routine is a function.
LibraryName	String	The name of the library that contains the class that contains the routine. The value is " " for system classes.
Name	String	The name of the routine including the argument datatypes and return value.
ObjectID	UnsignedLong	The internal ID for the object on which the routine is executing. The value is 0 for global and system functions.
ParentNode	TraceTreeNode	The parent of this node.

Functions

TraceTreeRoutine function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetChildrenList	ErrorReturn (enumerated)	Provides a list of the children (defined as TraceTreeNode objects) of this routine.

TraceTreeRoutine function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

TraceTreeUser object

The TraceTreeUser object provides information about a tree model node identified as an occurrence of an activity you selected for logging, including the activity argument and message. To access the extra properties of the TraceTreeUser object, you assign a TraceTreeNode object whose activity type is ActUser! to the TraceTreeUser object.

The TraceTreeUser object has no events.

Properties

TraceTreeUser property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActUser! which identifies the activity represented by the node as an occurrence of an activity you selected
Argument	Long	The argument passed to the TraceUser function
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control
Message	String	The message passed to the TraceUser function
ParentNode	TraceTreeNode	The parent of this node
TimerValue	Decimal	The timer value (in seconds) when the activity occurred

Functions

TraceTreeUser function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object (enumerated)	Returns the type of the object

TraceUser object

The TraceUser object provides information about a node in a trace file identified as an occurrence of an activity you selected for logging, including the activity argument and message. To access the extra properties of the TraceUser object, you assign a TraceActivityNode object whose activity type is ActUser! to the TraceUser object.

The TraceUser object has no events.

Properties

TraceUser property	Datatype	Description
ActivityType	TraceActivity (enumerated)	The value ActUser! which identifies the activity represented by the node as an occurrence of an activity you selected.
Argument	Long	The argument passed to the TraceUser function.
Category	TraceCategory (enumerated)	The value TraceAtomic! which indicates that the node is an activity that occurred in a single statement.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Message	String	The message passed to the TraceUser function.
TimerValue	Decimal	The timer value (in seconds) when the activity occurred.

Functions

TraceUser function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object (enumerated)	Returns the type of the object.

Transaction object

The Transaction object specifies the parameters that PowerBuilder uses to connect to a database.

You can customize your own version of the Transaction object by defining a class user object inherited from the built-in Transaction object.

For more information about creating a custom Transaction object, see the chapter on user objects in the *PowerBuilder Users Guide*.

For more information about using the Transaction object in an application, see *Application Techniques*.

Properties

Transaction property	Datatype	Description
AutoCommit	Boolean	The automatic commit indicator. Values are: TRUE – Commit automatically after every database activity. FALSE – Do not commit automatically after every database activity.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
Database	String	The name of the database with which you are connecting.
DBMS	String	PowerBuilder vendor identifier.
DBParm	String	DBMS-specific parameters.
DBPass	String	The password used to connect to the database.

Transaction property	Datatype	Description
Lock	String	The isolation level.
LogID	String	The name or ID of the user who logs in to the server.
LogPass	String	The password used to log in to the server.
ServerName	String	The name of the server on which the database resides.
SQLCode	Long	The success or failure code of the most recent operation. Return codes: 0 – Success 100 – Not found -1 – Error (use SQLDBCode or SQLErrText to obtain the details)
SQLDBCode	Long	The database vendor's error code.
SQLErrText	String	The database vendor's error message.
SQLNRows	Long	The number of rows affected (the database vendor supplies this number, so the meaning might not be the same in every DBMS)
SQLReturnData	String	DBMS-specific information
UserID	String	The name or ID of the user who will connect to the database

Events

Transaction event	Occurs
Constructor	When the user object is created
DBNotification	When an Oracle 10*g* server sends a notification that a database operation has occurred
DBError	When a database error occurs in the transaction
Destructor	When the user object is destroyed
SQLPreview	Occurs immediately before a SQL statement is submitted to the DBMS

Functions

Transaction function	Datatype returned	Description
ClassName	String	Returns the name assigned to the user object
DBHandle	Long	Returns the handle for your DBMS
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object

Transaction function	Datatype returned	Description
PostEvent	Boolean	Adds an event to the end of the message queue of the user object
SyntaxFromSQL	String	Generates DataWindow source code based on a SQL SELECT statement
TriggerEvent	Integer	Sends an event to the user object and executes the script associated with the event
TypeOf	Object	Returns the type of the user object

TransactionServer object

The TransactionServer object provides information about the current transaction context and enables a component running in a transaction server to control the transaction and its own life cycle.

Properties

TransactionServer property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control

Events

TransactionServer event	Occurs
Constructor	When the object is created
Destructor	When the object is destroyed

Functions

TransactionServer function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
CreateInstance	Integer	Creates an instance of a component running on the COM+ server. This function is called from within a component instance running on COM+.
DisableCommit	Integer	Declares that the component's transaction updates are inconsistent and cannot be committed in their present state.
EnableCommit	Integer	Declares that the component's work might be incomplete but its transaction updates are consistent and can be committed.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
ImpersonateClient	Integer	Allows the component to take on the security attributes of the client for the duration of a call.
IsCallerInRole	Integer	Indicates whether an object's direct caller is in a specified role (either individually or as part of a group).
IsImpersonating	Boolean	Queries whether the component is impersonating the client.
IsInTransaction (obsolete)	Boolean	Indicates whether the component is executing in a transaction.
IsSecurityEnabled	Boolean	Indicates whether or not security is enabled for the component
IsTransactionAborted (obsolete)	Boolean	Determines whether the current transaction has already been aborted.
Lookup (obsolete)	Long	Allows a component to create an instance of another component running on the same EAServer host. **Obsolete function** Lookup is an obsolete function, because EAServer is no longer supported since PowerBuilder 2017.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
RevertToSelf	Integer	If the component is impersonating the client, restores the component's security attributes.
SetAbort	Integer	Declares that the component cannot complete its work for the current transaction and that the transaction should be rolled back. The component instance will be deactivated when the method returns.
SetComplete	Integer	Declares that the transaction in which a component is participating should be committed and the component instance should be deactivated.
TriggerEvent	Integer	Triggers a specific event in the object and executes the script for the event.
TypeOf	Object	Returns the type of the object.

TransactionServer function	Datatype returned	Description
Which	Integer	Returns 0 if the object is not running in a transaction server, 1 if running in EAServer, or 2 if running in Microsoft MTS or IIS4.

TreeView control

A TreeView control is a hierarchical display of information. Each item in a TreeView control consists of text and pictures, which can be manipulated during program runtime.

Properties

TreeView property	Datatype	Description
Accelerator	Integer	Specifies the ASCII value of the accelerator key you want to use for the control.
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BackColor	Long	Specifies the numeric value of the background color: – 2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the control has a border. Values are: TRUE – Control has a border. FALSE – Control does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the border style of the control. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control is on top of other controls. FALSE – Control is not on top of other controls.

TreeView property	Datatype	Description
CheckBoxes	Boolean	Specifies whether the state images are replaced by check boxes. The check boxes are set to unchecked by default. The TreeView control processes mouse and keyboard input to toggle the checked state. Values are: TRUE – Check boxes are displayed. FALSE – Check boxes are not displayed. The state of an item's check box can be determined by checking the state picture index for the item: Unchecked = 1 Checked = 2
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DeleteItems	Boolean	Specifies whether the user can delete a TreeView item from a TreeView control using the Delete key. Values are: TRUE – The user can delete items from the control. FALSE – The user cannot delete items from the control.
DisableDragDrop	Boolean	Disable Drag Drop determines whether events for dragging, such as BeginDrag, are triggered when the user clicks on an item and drags. Values are: TRUE – Drag events are not triggered. FALSE – Drag events are triggered.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function. In either case, DisableDragDrop must be set to false for dragging to occur.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
EditLabels	Boolean	Specifies whether the user can edit the item labels in a control by clicking on a selected item. Values are: TRUE – The user can edit item labels. FALSE – The user cannot edit item labels.

TreeView property	Datatype	Description
Enabled	Boolean	Specifies whether the control is enabled (can be clicked). Values are: TRUE – Control can be clicked. FALSE – Control cannot be clicked.
FaceName	String	Specifies the name of the typeface in which the text of the control displays (for example, arial or courier).
FontCharSet	FontCharSet (enumerated)	Specifies the font character set used for the text in the control. The application must be running on an appropriate version of PowerBuilder under an operating system that supports the selected character set. For a complete list of possible values, see the list of properties for the FontCharSet variable on the Enumerated tab page of the Browser.
FontFamily	FontFamily (enumerated)	Specifies the font family (type style) used for the text in the control. Values are: AnyFont! Decorative! Modern! Roman! Script! Swiss!
FontPitch	FontPitch (enumerated)	Specifies the font pitch for the text in the control. Values are: Default! Fixed! Variable!
FullRowSelect	Boolean	Specifies whether full row selection is enabled. Values are: TRUE – Clicking anywhere on a row causes the entire row to be selected, and selecting any item in the row causes the entire row to be highlighted. FALSE – Selecting one item in a row does not cause the entire row to be highlighted or selected. This property cannot be used in conjunction with `HasLines = TRUE`.
HasButtons	Boolean	Specifies whether TreeView parent items have + and - buttons associated with them to indicate whether they are expanded (-) or collapsed (+). Values are: TRUE – Parent items have buttons. FALSE – Parent items do not have buttons. If SingleExpand is set to true to specify that only one item can be expanded, more than one item can be expanded by clicking on the item buttons if HasButtons is also true.

TreeView property	Datatype	Description
HasLines	Boolean	Specifies whether TreeView items are connected by lines. Values are: TRUE – Items are connected by lines. FALSE – Items are not connected by lines.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
HideSelection	Boolean	Specifies whether selected text stays selected (highlighted) even when the control does not have focus. Values are. TRUE – Text does not stay highlighted. FALSE – Text stays highlighted.
ImeMode	Integer	Specifies the input method editor mode. This property is relevant only to applications running on a Japanese version of PowerBuilder.
Indent	Integer	Specifies the size, in PowerBuilder units, that TreeView items are indented. Negative values are accepted but the items are outdented beyond the left edge of the control. Room is always reserved for the regular picture, whether or not it is displayed. An indent of less than 90 has no effect for the standard picture width. Set PictureWidth to 0 to remove extra space.
Italic	Boolean	Specifies whether the text in the control is italic. Values are: TRUE – Text is italic. FALSE – Text is not italic.
LayoutRTL	Boolean	Specifies that the layout of the control should be a mirror image of the standard layout. Values are: TRUE – Elements in the control are right justified FALSE – Elements in the control are left justified (default).
LinesAtRoot	Boolean	Specifies whether PowerBuilder will connect TreeView root items with lines. Values are: TRUE – TreeView control connects root items with lines when HasLines is also true. FALSE – Control does not connect root items with lines.
PictureHeight	Integer	Specifies the size, in pixels, for the height of the TreeView item picture. In a script, this value can be set only before a picture has been added to the picture index list. If the picture height is 0, PowerBuilder uses the height of the first picture added to the picture index list.
PictureMaskColor	Long	Specifies the color to be transparent when used in a TreeView item picture. Used when the picture is added at initialization or with the function AddPicture, and can be changed between adds.

TreeView property	Datatype	Description
PictureName[ ]	String	Specifies the names of the files containing the pictures added during initialization. The file extension *BMP*, *ICO*, *GIF*, *JPG* or *JPEG* is required. Not updated after initialization.
PictureWidth	Integer	Specifies the size, in pixels, for the width of the TreeView item picture. In a script, this value can be set only before a picture has been added to the picture index list. If the picture width is 0, PowerBuilder uses the width of the first picture added to the picture index list.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: TRUE – Characters display in right-to-left order. FALSE – Characters display in left-to-right order (default).
SingleExpand	Boolean	Specifies whether only the selected TreeView item is allowed to be expanded. A single mouse click selects an item. TRUE – When an item is selected, it is automatically expanded and the previously selected item is automatically collapsed. FALSE – More than one item can be expanded at a time. Note that if HasButtons = true, more than one item can be expanded at a time by clicking on the item buttons.
SortType	grSortType	Selects the sort method. Values are: Ascending! – Alphabetic by label. Descending! – Reverse-alphabetic by label. UserDefinedSort! – According to the script in the Sort event. Unsorted! – Not sorted. When SortType specifies sorting, sorting happens automatically. For Unsorted!, you can call functions for alphabetic sorting.
StatePictureHeight	Integer	Specifies the size, in pixels, for the height of the state picture. In a script, this value can be set only before a state picture has been added to the state picture list. If the state picture height is 0, PowerBuilder uses the height of the first picture added to the state picture index list.
StatePictureMaskColor	Long	Specifies the color to be transparent when used in a state picture. Used when the picture is added at initialization or with the function AddStatePicture, and can be changed between adds.

TreeView property	Datatype	Description
StatePictureName[ ]	String	Specifies the name of the picture used as the state picture. The state picture is displayed to the left of the regular picture. The item is shifted right to make room for it. The picture can be an icon, cursor, or bitmap supplied by the user or a stock picture from the PowerBuilder library. Not updated after initialization.
StatePictureWidth	Integer	Specifies the size in pixels for the width of the state picture. In a script, this value can be set only before a state picture has been added to the state picture list. If the state picture width is 0, PowerBuilder uses the width of the first picture added to the state picture index list.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
TextColor	Long	Specifies the numeric value of the color used for text: -2 to 16,777,215.
TextSize	Integer	Specifies the size of the text in the control, in points. For backward compatibility, the size is stored as a negative number; for example, 10-point text size is stored as -10.
ToolTips	Boolean	Specifies whether an item's label should be displayed in a tooltip if the label is cut off by the right edge of the control. Values are: TRUE – Tooltips displaying the label text are displayed when the label is cut off on the right side. This is the default. FALSE – Tooltips are not displayed.
TrackSelect	Boolean	Specifies whether items appear in a different color when the mouse moves over them (hot tracking). Values are: TRUE – An item changes color when the mouse moves over it. FALSE – An item does not change color when the mouse moves over or pauses on it.
Underline	Boolean	Specifies whether the text in the control is underlined. Values are: TRUE – Text is underlined. FALSE – Text is not underlined.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Weight	Integer	Specifies the stroke weight of the text in the control; for example, 400 for normal or 700 for bold.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.

TreeView property	Datatype	Description
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

TreeView event	Occurs
BeginDrag	When the user begins a drag operation with the left mouse button. If the DragAuto property is set to true, the drag begins automatically. If the DragAuto property is set to false, the drag operation must be done programmatically.
BeginLabelEdit	When the user starts to edit a TreeView item label. Return 1 to prevent setting to the new text. Return 0 to accept the new text.
BeginRightDrag	When the user begins a drag operation with the right mouse button. If the DragAuto property is set to true, the drag begins automatically. If the DragAuto property is set to false, the drag operation must be done programmatically.
Clicked	When the control is clicked.
Constructor	When the object is created, immediately before the Open event occurs in the window.
DeleteItem	When a TreeView item is deleted.
Destructor	When the object is destroyed, immediately after the Close event occurs in the window.
DoubleClicked	When the control is double-clicked.
DragDrop	When a dragged control is dropped on the TreeView control.
DragEnter	When a dragged control enters the control, including entering the narrow border around the display area.
DragLeave	When a dragged control leaves the control, including leaving by crossing into the tab page display area.
DragWithin	When a dragged control is within the control, but not on a TreeView item.
EndLabelEdit	When the user finishes editing a TreeView item label. Return 1 to prevent setting to the new text. Return 0 to accept the new text.
GetFocus	Just before the control receives focus (before it is selected and becomes active).
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
ItemCollapsed	When an item has collapsed.
ItemCollapsing	When an item is collapsing. Return 1 to prevent collapsing or 0 to allow it.
ItemExpanded	When an item has expanded.

TreeView event	Occurs
ItemExpanding	When an item is expanding. Return 1 to prevent expansion or 0 to allow it. If you want to populate the children each time an item expands, do it in the ItemExpanding event. If no children are created during the ItemPopulate or Item Expanding events, the item does not expand.
ItemPopulate	When an item is expanding for the first time. Return 1 to prevent expansion or 0 to allow it. If no children are created during the ItemPopulate or Item Expanding events, the item does not expand.
Key	When the user presses a key.
LoseFocus	When the control loses focus (becomes inactive).
Notify	When a TreeView control sends a wm_notify message to its parent.
Other	When a Windows message occurs that is not a PowerBuilder event.
RightClicked	When the control is right-clicked.
RightDoubleClicked	When the control is right double-clicked.
SelectionChanged	When the selection has changed.
SelectionChanging	When the selection is changing. Return 1 to prevent the selection from changing or 0 to allow it.
Sort	When sorting occurs and the SortType property is set to UserDefinedSort! The event occurs for each pair of items being sorted.

Functions

TreeView function	Datatype returned	Description
AddPicture	Integer	Adds an icon, cursor, or bitmap to the image list. Does not update PictureName.
AddStatePicture	Integer	Adds an icon, cursor, or bitmap to the state image list. Does not update StatePictureName.
ClassName	String	Returns the name of the control.
CollapseItem	Integer	Collapses the specified TreeView item.
DeleteItem	Integer	Deletes the specified TreeView item and all its children, if any.
DeletePicture	Integer	Deletes the specified icon, cursor, or bitmap from the image list.
DeletePictures	Integer	Deletes all icons, cursors, or bitmaps from the image list.
DeleteStatePicture	Integer	Deletes the specified icon, cursor, or bitmap from the state image list.
DeleteStatePictures	Integer	Deletes all icons, cursors, or bitmaps from the state image list.

TreeView function	Datatype returned	Description
Drag	Integer	Starts or ends the dragging of a TreeView item.
EditLabel	Integer	Starts the editing of a specific TreeView item label.
ExpandAll	Integer	Expands the children and subsequent levels for the specified TreeView item.
ExpandItem	Integer	Expands the specified TreeView item.
FindItem	Long	Returns the handle for the specified TreeView item.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetItem	Integer	Retrieves information for a specified item.
GetItemAtPointer	Integer	Gets the handle or the index of the item under the cursor.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Hides the specified TreeView item.
InsertItem	Long	Inserts a specified item at a specified position and level in a TreeView control.
InsertItemFirst	Long	Inserts a specified item as the first item at a specified level.
InsertItemLast	Long	Inserts a specified item as the last item at a specified level.
InsertItemSort	Long	Inserts a specified item at a specified level in the correct alphabetic position, if possible.
Move	Integer	Moves a control or object to a specified location.
PointerX	Integer	Determines the distance from the left edge of an object to the pointer location.
PointerY	Integer	Determines the distance from the top edge of an object to the pointer location.
PostEvent	Boolean	Adds the event to the end of the event queue of an object.
Print	Integer	Includes an object or lines of text in a print job.
Resize	Integer	Resizes a control to the specified dimensions.
SelectItem	Integer	Highlights an item in the control, making it the current item.
SetDropHighlight	Integer	Designates the specified TreeView item as the target of a DragDrop operation.
SetFirstVisible	Integer	Sets the specified TreeView item as the first item visible in a TreeView control. If there are enough items to allow it, the specified item scrolls to the top of the control. If not, it is selected.
SetFocus	Integer	Sets the focus for a specified object or control.
SetItem	Integer	Sets the information for the specified TreeView item.
SetLevelPictures	Integer	Specifies a picture index for all TreeView items at a specific level in a TreeView control.

TreeView function	Datatype returned	Description
SetOverlayPicture	Integer	Maps a picture index to an overlay picture index. Only four overlay picture indexes are available.
SetPosition	Integer	Sets the position of the TreeView control in the front-to-back order within a window.
SetRedraw	Integer	Controls the automatic redraw of an object after its properties have changed.
Show	Integer	Makes an object or control visible, if it is hidden. If the object is already visible, Show brings it to the top.
Sort	Integer	Sorts the children of a specified TreeView item according to the method of the SortType property or alphabetically if SortType is Unsorted!.
SortAll	Integer	Sorts the children of an item and all subsequent levels according to the method of the SortType property or alphabetically if SortType is Unsorted!.
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

TreeViewItem object

A TreeViewItem is a system structure that populates the properties for individual items in a TreeView control. A TreeViewItem has no events.

Properties

TreeViewItem property	Datatype	Description
Bold	Boolean	Specifies whether the item is bold. Values are: TRUE – The item is bold. FALSE – The item is not bold.

TreeViewItem property	Datatype	Description
Children	Boolean	Specifies whether the item has children. Values are: TRUE – The item has children. FALSE – The item does not have children. You can use this property to make the TreeView behave as though it has children, even when it does not. You can do this to get ItemPopulate and ItemExpanding events when the item does not yet have any children.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
CutHighLighted	Boolean	Specifies whether the item is the target of a cut operation. Values are: TRUE – The item is the target of a cut operation. FALSE – The item is not the target of a cut operation.
Data	Any	Assigns any user-defined data to a TreeView item.
DropHighLighted	Boolean	Specifies whether the item is the target of a DragDrop operation. Values are: TRUE – The item is the target of a DragDrop operation. FALSE – The item is not the target of a DragDrop operation.
Expanded	Boolean	Specifies whether the item is expanded. Values are: TRUE – The item is expanded. FALSE – The item is not expanded.
ExpandedOnce	Boolean	Specifies whether the item has been expanded at least once, also meaning the item has been populated with children. Values are: TRUE – The item has been expanded once. FALSE – The item has not been expanded once.
HasFocus	Boolean	Specifies whether the item has focus. Values are: TRUE – The item has focus. FALSE – The item does not have focus.
ItemHandle	Long	Identifies the handle associated with the item.
Label	String	Identifies the string label associated with the item.
Level	Integer	Indicates the level of the item in the TreeView control.
OverlayPictureIndex	Integer	Identifies the overlay picture associated with the item. The overlay picture is displayed on top of the item's picture. If 0, no overlay is displayed.
PictureIndex	Integer	Identifies the picture displayed to the left of the item label. If 0, no picture appears and the space specified by the TreeView's PictureWidth property is blank.
SelectedPictureIndex	Integer	Identifies the picture associated with the item when it is selected. If 0, no picture is displayed when selected.

TreeViewItem property	Datatype	Description
Selected	Boolean	Specifies whether the item is selected. Values are: TRUE – The item is selected. FALSE – The item is not selected.
StatePictureIndex	Integer	Identifies the state picture associated with the item. The state picture appears to the left of the regular picture. If 0, no state picture appears and no space is reserved for the picture.

Functions

TreeViewItem function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
TypeOf	Object	Returns the type of the object.

TypeDefinition object

TypeDefinition is used in the VariableDefinition class.

Information about the type of a variable. The variable can be a single value, an object, or an array. TypeDefinition is an abstract class that is the ancestor of ClassDefinition, SimpleTypeDefinition, and EnumerationDefinition. It has no events.

Properties

TypeDefinition property	Datatype	Description
Category	TypeCategory	Specifies whether the type is simple, enumerated, or a class or structure. Values are: SimpleType! EnumeratedType! ClassOrStructureType!

TypeDefinition property	Datatype	Description
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DataTypeOf	String	The system class name or simple datatype of the variable. DataTypeOf is a string representation of a value of the Object enumerated datatype. Values are lowercase with no exclamation point. Sample values include: window string any dropdownlistbox For objects you have defined, the datatype is the system class from which your object is inherited.
IsStructure	Boolean	Indicates whether the type is a structure.
IsSystemType	Boolean	Indicates whether the type is defined by PowerBuilder as opposed to a type defined in a PBL by a user.
IsVariableLength	Boolean	Specifies whether the datatype has a fixed size. Values are: TRUE – The datatype is variable length, meaning the datatype is a string, any, blob, or unbounded array. FALSE – The datatype is a fixed length.
IsVisualType	Boolean	Indicates whether the type is a visual (displayable) or nonvisual type. Values are: TRUE – The type is visual, for example, a window or a control. FALSE – The type is non-visual, for example, a class user object or a simple datatype.
LibraryName	String	The fully qualified name of the library the type was loaded from. The library might no longer contain the type. If a program manipulates the contents of libraries, its class could have been moved or deleted after it was loaded.
Name	String	The name of the type. For a nested type, the name is returned in the form of *libraryEntryName`typeName*

Functions

TypeDefinition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object

TypeDefinition function	Datatype returned	Description
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

ULSync object

The ULSync object is derived from the MLSynchronization base class but is not supported in the current release.

UserObject object

UserObjects are custom visual objects that you can build to supplement the standard PowerBuilder objects. UserObjects can display information, request information from a user, and respond to mouse or keyboard actions. You can also create a TabPage UserObject. Use the User Object painter to build UserObjects.

When you place a visible UserObject in a window, you are actually placing a UserObject *control* in the window. The control holds an instance of the UserObject you select for the window.

Properties

UserObject property	Datatype	Description
BackColor	Long	Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.
Border	Boolean	Specifies whether the object has a border. Values are: TRUE – Object has a border. FALSE – Object does not have a border.
BorderStyle	BorderStyle (enumerated)	Specifies the style of the border of the object. Values are: StyleBox! StyleLowered! StyleRaised! StyleShadowBox!
BringToTop	Boolean	Specifies whether PowerBuilder moves the object to the top of the front-to-back order of the window. Values are: TRUE – Object moved to top. FALSE – Object not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

UserObject property	Datatype	Description
ClassName	String	(External user objects only) Returns the name assigned to the object.
ColumnsPerPage	Integer	Specifies the number of columns on a scroll page. The default is 0 (10 columns per page). For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerColumn by ColumnsPerPage to determine how many PowerBuilder units to scroll the object horizontally when the user clicks in the scroll bar.
Control[ ]	WindowObject	Specifies the control's objects. You cannot change the contents of this array in a script.
DragAuto	Boolean	Specifies whether PowerBuilder puts the object automatically into Drag mode. Values are: TRUE – When the object is clicked, it is automatically in Drag mode. FALSE – When the object is clicked, it is not automatically in Drag mode. You have to manually put the object into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the object (the *ICO* file). The default icon is a box the size of the object. When the user drags the object, the icon displays when the object is over an area in which the object can be dropped (a valid drop area). When the object is over an area that is not a valid drop area, the No-Drop icon displays.
Enabled	Boolean	Specifies whether the object is enabled (can be selected). Values are: TRUE – Object can be selected. FALSE – Object cannot be selected.
Height	Integer	Specifies the height of the object, in PowerBuilder units.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays. Values are: TRUE – Horizontal scroll bar displays. FALSE – Horizontal scroll bar does not display.
LibraryName	String	(External user objects only) The name of the dynamic-link library (DLL) that contains an external user object class.
LinesPerPage	Integer	Specifies the number of lines on a page. The default is 0 (10 lines per page). For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerLine by LinesPerPage to determine how many PowerBuilder units to scroll the object vertically when the user clicks in the scroll bar.

UserObject property	Datatype	Description
ObjectType	UserObjects (enumerated)	Specifies the type of user object. Valid values are: CustomVisual! ExternalVisual!
PictureMaskColor	Long	Specifies the numeric value of the color in the picture that is changed to the background color. Values can be: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property applies only when PictureName is a bitmap and only when the UserObject is a tab page.
PictureName	String	Specifies a value of the Pointer enumerated datatype or the filename of the bitmap, cursor, or icon or to be displayed on the tab. This property applies only when the UserObject is a tab page.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the object.
PowerTipText	String	Specifies text to be displayed as a PowerTip for the tab when the Tab control's PowerTips property is true. This property applies only when the UserObject is a tab page.
Style	Long	Specifies any additional style bits you want to use to control how the object displays (external user object only).
TabBackColor	Long	Specifies the numeric value of the tab background color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property applies only when the UserObject is a tab page. It is not supported on Windows XP because the current XP theme controls the appearance of the tab.
TabTextColor	Long	Specifies the numeric value of the tab text color: -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*. This property applies only when the UserObject is a tab page.
TabOrder	Integer	Specifies tab value of the control within the user object (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the object.
Text	String	Specifies the text that displays in the object.

UserObject property	Datatype	Description
UnitsPerColumn	Integer	Specifies the number of PowerBuilder units to be scrolled right or left when a user clicks the left or right arrow in the horizontal scroll bar in a window or user object. The default is 0 (1/100 of the width of the window). To make the end of the scroll bar match the content, UnitsPerLine must be set according to the content width. For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerColumn by ColumnsPerPage to determine the number of PowerBuilder units to scroll the window horizontally when the user clicks in the scroll bar.
UnitsPerLine	Integer	Specifies the number of PowerBuilder units to be scrolled up or down when a user clicks the up or down arrow in the vertical scroll bar in a window or user object. The default is 0 (1/100 of the window height). To make the end of the scroll bar match the content, UnitsPerLine must be set according to the content length. For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerLine by LinesPerPage to determine the number of PowerBuilder units to scroll the window vertically when the user clicks in the scroll bar.
Visible	Boolean	Specifies whether the object is visible. Values are: TRUE – Object is visible. FALSE – Object is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar displays. Values are: TRUE – Vertical scroll bar displays. FALSE – Vertical scroll bar does not display.
Width	Integer	Specifies the width of the object, in PowerBuilder units.
X	Integer	Specifies the X position (distance from the left edge of screen) of the object, in PowerBuilder units.
Y	Integer	Specifies the Y position (distance from the top of screen) of the object, in PowerBuilder units.

Events

UserObject event	Occurs
Constructor	Immediately before the Open event occurs in the window

UserObject event	Occurs
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the object
DragEnter	When a dragged control enters the object
DragLeave	When a dragged control leaves the object
DragWithin	When a dragged control is within the object
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the object

Functions

UserObject function	Datatype returned	Description
AddItem	Integer	Adds item to list.
ClassName	String	Returns the name assigned to the object.
CloseUserObject	Integer	Removes the specified user object from view, closes it, and executes its Destructor event.
CreatePage	Integer	Creates a tab page if it has not already been created.
DeleteItem	Integer	Deletes item from list.
Drag	Integer	Starts or ends the dragging of the object.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the object invisible.
InsertItem	Integer	Inserts item in list.
Move	Integer	Places the object in a new location specified by the X and Y arguments.
OpenUserObject	Integer	Displays the specified user object, making its properties available to scripts.
OpenUserObjectWithParm	Integer	Displays the specified user object, making its properties available to scripts, and stores a parameter in the system's Message object.
PageCreated	Boolean	Reports whether a tab page has been created.
PointerX	Integer	Returns the distance from the left edge of the screen to the pointer, in PowerBuilder units.

UserObject function	Datatype returned	Description
PointerY	Integer	Returns the distance from the top of the screen to the pointer, in PowerBuilder units.
PostEvent	Boolean	Adds an event to the end of the message queue for the object.
Print	Integer	Prints the object.
Resize	Integer	Changes the size of the object based on the width and height.
SetFocus	Integer	Sets focus to the object.
SetPosition	Integer	Specifies the position of the object in the front-to-back order of the window.
SetRedraw	Integer	Turns on or off automatic redrawing of the object after every change.
Show	Integer	Makes the object visible.
TriggerEvent	Integer	Sends an event to the object and executes the script associated with the event.
TypeOf	Object	Returns the type of the object.

VariableCardinalityDefinition object

A class that provides information about the cardinality of a variable. It reports whether the associated variable is a single instance or an array. If it is an array, you can get information about the dimensions. VariableCardinalityDefinition is used in the VariableDefinition object. It has no events.

Properties

VariableCardinality Definition property	Datatype	Description
ArrayDefinition[ ]	ArrayBounds	When the associated variable's Cardinality is BoundedArray!, an array with an ArrayBounds object for each dimension in the array being described. When Cardinality is UnboundedArray!, ArrayDefinition has a single ArrayBounds object with LowerBound and UpperBound properties both set to 0. The extent of the array is not part of the class definition. Not valid when Cardinality is ScalarType!

VariableCardinality Definition property	Datatype	Description
Cardinality	Variable CardinalityType	The cardinality of the associated variable. Values are: ScalarType! UnboundedArray! BoundedArray!
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

Functions

VariableCardinality Definition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

VariableDefinition object

A class describing the characteristics of a variable, property, or argument. VariableDefinition is used as a property in the ClassDefinition and ScriptDefinition objects. It has no events.

You cannot start with a variable in your application and get a VariableDefinition object for it. Instead, you access the VariableDefinition instances that are elements of the VariableList array of a ClassDefinition instance or the ArgumentList array of a ScriptDefinition instance.

The VariableDefinition object has information about:

- The variable's name and type
- Whether the variable is a scalar or an array and information about the array
- The variable's initial value, whether the value overrides an ancestor's value, and whether the variable is a constant
- The read and write access levels for the variable

- The scope of the variable (global, shared, instance, local, argument), including whether the variable is an argument and how the argument is passed

Properties

VariableDefinition property	Datatype	Description
CallingConvention	ArgCalling Convention	The way an argument is passed when Kind is VariableArgument! Values are: ByReferenceArgument! ByValueArgument! ReadOnlyArgument! VarListArgument! VarListArgument! applies only to arguments for built-in PowerBuilder functions. They are shown as ellipses in the Browser. For an example, see the ImportString function for DataWindow controls.
Cardinality	Variable Cardinality Definition	Cardinality information for the variable.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
InitialValue	Any	The initial value of the variable. Not valid when Kind is VariableArgument!.
IsConstant	Boolean	Indicates whether the variable is a constant. Not valid when Kind is VariableArgument!.
IsControl	Boolean	Indicates whether the variable is a control defined as a nested class within its parent, rather than an instance variable with a control class as its datatype. Valid only when Kind is VariableInstance!.
IsUserDefined	Boolean	Indicates whether the variable is a user-defined variable, instead of a property or variable defined by PowerBuilder. Always true for local variables. True for arguments if the function was also user-defined.

VariableDefinition property	Datatype	Description
Kind	VariableKind	The scope of the variable. Values are: VariableGlobal! VariableShared! VariableInstance! VariableArgument! VariableLocal! Global variables are found only in the Application object. Argument and local variables are found only in scripts.
Name	String	The name of the variable.
OverridesAncestor Value	Boolean	Indicates whether the current initial value overrides an ancestor's initial value. Valid only when Kind is VariableInstance!
ReadAccess	VarAccess	The read access to the variable. Values are: Private! Public! Protected! System! Not valid when Kind is VariableArgument! or VariableLocal!
TypeInfo	TypeDefinition	Type information for the variable.
WriteAccess	VarAccess	The write access to the variable. Values are: Private! Public! Protected! System! Not valid for VariableArgument! or VariableLocal!

Functions

VariableDefinition function	Datatype returned	Description
ClassName	String	Returns the name assigned to the object
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
TypeOf	Object	Returns the type of the object

VProgressBar control

You can use a progress bar to indicate the progress of a lengthy operation, such as an installation program that copies a large number of files. The VProgressBar control is a vertical rectangle that fills with the system highlight color as the operation progresses.

Properties

VProgressBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.

VProgressBar property	Datatype	Description
MaxPosition	Unsigned Integer	Specifies the value of the Position property when the progress bar is at the top of the control. This value can be different from the end of the control's range, set with the SetRange function. The default value is 100.
MinPosition	Unsigned Integer	Specifies the value of the Position property when the progress bar is at the bottom of the control. This value can be different from the start of the control's range, set with the SetRange function. The default value is 0.
Pointer	String	Specifies the name of the stock pointer or file containing the pointer used for the control.
Position	Integer	Specifies the value of the current position within the range of the control (set with the SetRange function). The control uses the range and the current position to determine the percentage of the progress bar to fill with the highlight color.
SetStep	Integer	Specifies a step increment for the progress bar. The default is 10.
SmoothScroll	Boolean	Specifies that the control displays as a smooth scrolling bar instead of the default segmented bar.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

VProgressBar event	Occurs
Clicked	When the left mouse button is pressed on the control
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DoubleClicked	When the left mouse button is double-clicked on the control
DragDrop	When a dragged control is dropped on the control

VProgressBar event	Occurs
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LoseFocus	When the control loses focus (becomes inactive)
Other	When a Windows message occurs that is not a PowerBuilder event
RButtonDown	When the right mouse button is pressed on the control

Functions

VProgressBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
OffsetPos	Integer	Moves the control's current position by the amount specified
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets the focus to the control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRange	Integer	Sets the range of the control. The control uses the range and the current position to determine the percentage of the progress bar to fill with the highlight color
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible

VProgressBar function	Datatype returned	Description
StepIt	Integer	Moves the control's current position by the amount specified by the value of the SetStep property
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

VScrollBar control

A VScrollBar is a vertical bar with arrows at either end and a scroll box. Typically, you use a VScrollBar control as a slider control for users to specify a value on a continuous scale, or as a way to display information graphically to the user.

Usage note
The VScrollBar control is not the vertical scroll bar that displays to allow the user to scroll through information in a control or window.

Properties

VScrollBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order of the window. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.

VScrollBar property	Datatype	Description
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
MaxPosition	Integer	Specifies the value of the Position property when the scroll box is at the bottom of the scroll bar.
MinPosition	Integer	Specifies the value of the Position property when the scroll box is at the top of the scroll bar.
Pointer	String	Specifies the name of the stock pointer or the file containing the pointer used for the control.
Position	Integer	Specifies the value between MinPosition and MaxPosition that indicates the position of the scroll box.
StdWidth	Boolean	Specifies whether the standard scroll bar width is used for the VScrollBar. Values are: TRUE – Standard width used. FALSE – Standard width not enforced for the control.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

VScrollBar event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LineDown	When the down arrow of the control is clicked
LineUp	When the up arrow of the control is clicked
LoseFocus	When the control loses focus (becomes inactive)
Moved	When the scroll box is moved (use the Position property to determine the new location)
Other	When a Windows message occurs that is not a PowerBuilder event.
PageDown	When the open space below the scroll box is clicked
PageUp	When the open space above the scroll box is clicked
RButtonDown	When the right mouse button is pressed on the control

Functions

VScrollBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
Drag	Integer	Starts or ends the dragging of the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
Hide	Integer	Makes the control invisible
Move	Integer	Moves the control to a specified location
PointerX	Integer	Returns the distance of the pointer from the left edge of the control
PointerY	Integer	Returns the distance of the pointer from the top of the control
PostEvent	Boolean	Adds an event to the end of the message queue for the control
Print	Integer	Prints the control

VScrollBar function	Datatype returned	Description
Resize	Integer	Changes the size of the control
SetFocus	Integer	Sets focus to the specified control
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties
Show	Integer	Makes the control visible
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

VTrackBar control

Like a scroll bar, a trackbar is used as a scrolling control, but clicking on the trackbar slider moves it in discrete increments instead of continuously. The VTrackBar control has a series of tick marks to the right of the trackbar channel.

To enable this control to be used properly from the keyboard, you must add code to the LineDown, LineUp, PageDown, and PageUp events. The code you add should change the slider Position property by the appropriate value and then pass the new slider position to the object or objects you associate with the trackbar control. You must code the Moved event if you want the trackbar control to pass on the slider position after the slider is dragged with a mouse.

Usage note
Use a trackbar when you want the user to select a discrete value. For example, you might use a trackbar to enable a user to select a timer interval or the size of a window.

Properties

VTrackBar property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
BringToTop	Boolean	Specifies whether PowerBuilder moves the control to the top of the front-to-back order. Values are: TRUE – Control moved to top. FALSE – Control not moved to top.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
DragAuto	Boolean	Specifies whether PowerBuilder puts the control automatically into Drag mode. Values are: TRUE – When the control is clicked, the control is automatically in Drag mode. FALSE – When the control is clicked, the control is not automatically in Drag mode. You have to manually put the control into Drag mode by using the Drag function.
DragIcon	String	Specifies the name of the stock icon or the file containing the icon you want to display when the user drags the control (the *ICO* file). The default icon is a box the size of the control. When the user drags the control, the icon displays when the control is over an area in which the control can be dropped (a valid drop area). When the control is over an area that is not a valid drop area, the No-Drop icon displays.
Height	Integer	Specifies the height of the control, in PowerBuilder units.
LineSize	Integer	Specifies how far the slider moves in response to keyboard input from the arrow keys.
MaxPosition	Integer	Specifies the value of the Position property when the slider is at the bottom of the control.
MinPosition	Integer	Specifies the value of the Position property when the slider is at the top of the control.
PageSize	Integer	Specifies how far the slider moves in response to keyboard or mouse input. Setting PageSize to 1 indicates moving 1 increment in the range of values.
Pointer	String	Specifies the name of the stock pointer or file containing the pointer used for the control.

VTrackBar property	Datatype	Description
Position	Integer	Specifies a value between MinPosition and MaxPosition specifying the position of the slider.
Slider	Boolean	Specifies whether or not the trackbar contains a slider.
SliderSize	Integer	Specifies the size of the slider on the trackbar.
TabOrder	Integer	Specifies the tab value of the control within the window (0 means the user cannot tab to the control).
Tag	String	Specifies the tag value assigned to the control.
TickFrequency	Integer	Specifies tick mark frequency. Setting TickFrequency to 1 indicates 1 tick mark for each increment in the trackbar range of values.
TickMarks	VTickMarks (enumerated)	Specifies where tickmarks should be displayed. Values are: VTicksOnRight! VTicksOnLeft! VTicksOnBoth! VTicksOnNeither!
Visible	Boolean	Specifies whether the control is visible. Values are: TRUE – Control is visible. FALSE – Control is not visible.
Width	Integer	Specifies the width of the control, in PowerBuilder units.
X	Integer	Specifies the X position (the distance from the left edge of the window), in PowerBuilder units.
Y	Integer	Specifies the Y position (the distance from the top of the window), in PowerBuilder units.

Events

VTrackBar event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window
DragDrop	When a dragged control is dropped on the control
DragEnter	When a dragged control enters the control
DragLeave	When a dragged control leaves the control
DragWithin	When a dragged control is within the control
GetFocus	Just before the control receives focus (before it is selected and becomes active)
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control
LineDown	When the down arrow key is clicked

VTrackBar event	Occurs
LineUp	When the up arrow key is clicked
LoseFocus	When the control loses focus (becomes inactive)
Moved	When the slider is moved (use the Position property to determine the new location)
Other	When a Windows message occurs that is not a PowerBuilder event
PageDown	When the Page Down key is clicked or when mouse clicks are made below the slider in the trackbar channel
PageUp	When the Page Up key is clicked or when mouse clicks are made above the slider in the trackbar channel
RButtonDown	When the right mouse button is pressed on the control

Functions

VTrackBar function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control.
Drag	Integer	Starts or ends the dragging of the control.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetParent	PowerObject	Returns a reference to the name of the parent object.
Hide	Integer	Makes the control invisible.
Move	Integer	Moves the control to a specified location.
PointerX	Integer	Returns the distance of the pointer from the left edge of the control.
PointerY	Integer	Returns the distance of the pointer from the top of the control.
PostEvent	Boolean	Adds an event to the end of the message queue for control.
Print	Integer	Prints the control.
Resize	Integer	Changes the size of the control.
SelectionRange	Integer	Sets a selection range for the trackbar. When you select a range, a blue line is drawn in the channel of the trackbar and two arrows are drawn where the tickmarks are placed to indicate the beginning and end of the selection range.
SetFocus	Integer	Sets the focus to the control.
SetPosition	Integer	Specifies the position of the control in the front-to-back order of the window.
SetRedraw	Integer	Controls automatic redrawing of the control after each change in its properties.
Show	Integer	Makes the control visible.

VTrackBar function	Datatype returned	Description
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event.
TypeOf	Object	Returns the type of the control.

Window control

Windows are the main interface between the user and a PowerBuilder application. Windows can display information, request information from a user, and respond to the user's mouse or keyboard actions.

The definition of a window includes properties, events, and functions. The properties determine the style of the window—how it looks. The events are actions in the window; when an event is triggered, the associated script is executed. The functions can trigger events in the window, manipulate or change the window, or provide information about the window.

Properties

Every window has a style that determines how it looks to the user. That style is governed by values assigned to the properties of the window.

Window property	Datatype	Description
AccessibleDescription	String	A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users.
AccessibleName	String	A label that briefly describes the control, such as the text in a button or the name of a menu item.
AccessibleRole	AccessibleRole (enumerated)	Describes what kind of user interface element the control is.
AnimationTime	Integer	Specifies how long an animation specified with the OpenAnimation or CloseAnimation property plays. Value is a positive integer in the range 1 to 5000 milliseconds. The default is 200 milliseconds.
BackColor	Long	Specifies the numerical value of the background color of the window. Values are -2 to 16,777,215. For more information about color, see the RGB function in the *PowerScript Reference*.

Window property	Datatype	Description
Border	Boolean	Specifies whether the window has a border. Values are: •TRUE – Has a border. •FALSE – Does not have a border.
BringToTop	Boolean	Specifies whether PowerBuilder moves the window to the top of the front-to-back order. Values are: •TRUE – Moves to the top. •FALSE – Does not move to the top.
Center	Boolean	Causes the window to be centered when it is created or sized. Values are: •TRUE – Window is centered. •FALSE – Window is not centered.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClientEdge	Boolean	Specifies whether the client area of the window appears sunken within the frame. •TRUE – Client area appears sunken. •FALSE – Client area does not appear sunken.
CloseAnimation	WindowAnimation Style (enumerated)	Specifies an optional animation effect that displays when the window closes.
ColumnsPerPage	Integer	Specifies the number of columns on a page. The default is 0 (10 columns per page). For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerColumn by ColumnsPerPage to determine the number of PowerBuilder units to scroll the window horizontally when the user clicks in the scroll bar.
ContextHelp	Boolean	When WindowType = Response!, this property specifies whether the small question mark button appears in the title bar. The question mark button can fire the Help event on the control that is clicked next. •TRUE – Question mark button is displayed in title bar next to minimize button in Response windows. •FALSE – Question mark button is not displayed.
Control[ ]	WindowObject	Contains the controls in the window. You should not change the contents of this array in a script.
ControlMenu	Boolean	Specifies whether the Control Menu box displays in the title bar. Values are: •TRUE – Displays in title bar. •FALSE – Does not display in title bar.

Window property	Datatype	Description
Enabled	Boolean	Specifies whether the window is enabled (can send and receive messages). Values are: •TRUE – Can send/receive messages. •FALSE – Cannot send/receive messages.
Height	Integer	Specifies the height of the window, in PowerBuilder units. You cannot resize minimized or maximized windows at runtime.
HScrollBar	Boolean	Specifies whether a horizontal scroll bar displays in the window. Values are: •TRUE – A scroll bar displays. •FALSE – A scroll bar does not display.
Icon	String	Specifies a stock icon or an ICO file that displays when the window is minimized. The default value is AppIcon!, which is the icon selected for the Application object. If no icon is selected for the Application object, the Windows logo is used.
LinesPerPage	Integer	Specifies the number of lines on a page. The default is 0 (10 lines per page). For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerLine by LinesPerPage to determine the number of PowerBuilder units to scroll the window vertically when the user clicks in the scroll bar.
MaxBox	Boolean	Specifies whether a Maximize Box displays in the title bar. Values are: •TRUE – Maximize Box displays. •FALSE – Maximize Box does not display.
MenuID	Menu	Specifies the ID of a menu. PowerBuilder uses MenuID internally. To change the menu for a window from a script, use the ChangeMenu function; to display a pop-up menu, use the PopMenu function. In both functions, enter the fully qualified name to identify the menu or Menu object.
MenuName	String	Specifies the name of a menu. PowerBuilder uses MenuName internally. To change the menu for a window from a script, use the ChangeMenu function; to display a pop-up menu, use the PopMenu function. In both functions, enter the fully qualified name to identify the menu or Menu object.
MinBox	Boolean	Specifies whether a Minimize Box displays in the title bar. Values are: •TRUE – Minimize Box displays. •FALSE – Minimize Box does not display.

Window property	Datatype	Description
OpenAnimation	WindowAnimation Style (enumerated)	Specifies an optional animation effect that displays when the window opens.
PaletteWindow	Boolean	When WindowType = Popup!, this property specifies that the window has an appearance appropriate for small palette windows that display over the application. •TRUE – Pop-up window displays as the topmost window with a smaller Close button in the title bar and no Minimize or Maximize buttons. •FALSE – No change in appearance.
Pointer	String	Specifies the name of the file containing the pointer that is used for the window.
Resizable	Boolean	Specifies whether the window is resizable. Values are: •TRUE – Window is resizable. •FALSE – Window is not resizable.
RightToLeft	Boolean	Specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are: •TRUE – Characters display in right-to-left order. •FALSE – Characters display in left-to-right order.
Tag	String	Specifies the tag value assigned to the window.
Tabs	Enumerated	**Properties for the shapes of the tabs:** •windowdocktabslanted! •windowdocktabrectangular! •windowdocktabsingleslanted! **Location of the close button on a tab, if any:** •windowdocktabclosebuttonnone! •windowdocktabclosebuttononactive! •windowdocktabclosebuttonshared! **Colors of tabs:** •gradients are available •default to theme colors is available **Icon and Scroll Button for tabbed windows or documents:** •TabbedWindowTabIcon / TabbedDocumentTabIcon •TabbedWindowTabScroll / TabbedDocumentTabScroll

Window property	Datatype	Description
Tabbed Window and Document Title Bars	Enumerated	**Colors of tabbed window and document title bars:** •TabbedWindowActiveTabBackColor / TabbedDocumentActiveTabBackColor •TabbedWindowActiveTabGradientBackColor / TabbedDocumentActiveTabGradientBackColor •TabbedWindowActiveTabTextColor / TabbedDocumentActiveTabTextColor •TabbedWindowInActiveTabBackColor / TabbedDocumentInActiveTabBackColor •TabbedWindowInActiveTabGradientBackColor / TabbedDocumentInActiveTabGradientBackColor •TabbedWindowInActiveTabTextColor / TabbedDocumentInActiveTabTextColor •TabbedWindowMouseoverTabBackColor / TabbedDocumentMouseoverTabBackColor •TabbedWindowMouseoverTabGradientBackColor / TabbedDocumentMouseoverTabGradientBackColor •TabbedWindowMouseoverTabTextColor / TabbedDocumentMouseoverTabTextColor •TabbedWindowTabsAreaColor / TabbedDocumentTabsAreaColor •TabbedWindowTabsAreaGradientColor / TabbedDocumentTabsAreaGradientColor •TabbedWindowTabsAreaGradientVert / TabbedDocumentTabsAreaGradientVert **Title bar states:** •TitleBarActiveColor / TitleBarInActiveColor •TitleBarActiveGradientColor / TitleBarInActiveGradientColor •TitleBarActiveGradientVert / TitleBarInActiveGradientVert •TitleBarActiveaTextColor / TitleBarInActiveTextColor
Title	String	Specifies the text of the window title.
TitleBar	Boolean	Specifies whether a title bar displays. Values are: •TRUE – Title bar displays. •FALSE – No title bar displays. The user can move a window only if it has a title bar.

Window property	Datatype	Description
ToolbarAlignment	Toolbar Alignment (enumerated)	In an MDI frame window, specifies where the toolbar displays. Values are: •AlignAtBottom! •AlignAtLeft! •AlignAtRight! •AlignAtTop! •Floating!
ToolbarHeight	Integer	In an MDI frame window, specifies the height of the toolbar when it is a floating toolbar.
ToolbarVisible	Boolean	In an MDI frame window, specifies whether the toolbar displays. Values are: •TRUE – Toolbar displays. •FALSE – Toolbar does not display.
ToolbarWidth	Integer	In an MDI frame window, specifies the width of the toolbar when it is a floating toolbar.
ToolbarX	Integer	In an MDI frame window, specifies the X coordinate (distance from the left edge of the window, in PowerBuilder units) of the toolbar when it is a floating toolbar.
ToolbarY	Integer	In an MDI frame window, specifies the Y coordinate (distance from the top of the window, in PowerBuilder units) of the toolbar when it is a floating toolbar.
Transparency	Integer	Specifies the transparency of a window. Value is a percentage in the range 0 (opaque) to 100 (completely transparent).
UnitsPerColumn	Integer	Specifies the number of PowerBuilder units to be scrolled right or left when a user clicks the left or right arrow in the horizontal scroll bar in a window or user object. The default is 0 (1/100 of the width of the window). To make the end of the scroll bar match the content, UnitsPerLine must be set according to the content width. For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerColumn by ColumnsPerPage to determine the number of PowerBuilder units to scroll the window horizontally when the user clicks in the scroll bar.

Window property	Datatype	Description
UnitsPerLine	Integer	Specifies the number of PowerBuilder units to be scrolled up or down when a user clicks the up or down arrow in the vertical scroll bar in a window or user object. The default is 0 (1/100 of the window height). To make the end of the scroll bar match the content, UnitsPerLine must be set according to the content length. For information, see Scrolling in windows and user objects on page 591. PowerBuilder multiplies UnitsPerLine by LinesPerPage to determine the number of PowerBuilder units to scroll the window vertically when the user clicks in the scroll bar.
Visible	Boolean	Specifies whether the window is visible. Values are: •TRUE – Window is visible. •FALSE – Window is not visible.
VScrollBar	Boolean	Specifies whether a vertical scroll bar displays. Values are: •TRUE – Scroll bar displays. •FALSE – Scroll bar does not display.
Width	Integer	Specifies the width of the window, in PowerBuilder units. You cannot resize minimized or maximized windows at runtime.
WindowDockOptions	WindowDockOptions (enumerated)	WindowDockOptions are for child windows to specify how they can be opened: •WindowDockOptionAll! •WindowDockOptionTabbedDocumentOnly! •WindowDockOptionDockedOnly! •WindowDockOptionFloatOnly! •WindowDockOptionTabbedDocumentAndDockedOnly! •WindowDockOptionTabbedDocumentAndFloatOnly! •WindowDockOptionDockedAndFloatOnly!
WindowDockState	WindowDockState (enumerated)	Specifies the docking behaviors for the sheets that open in the MDI (multiple document interface) frame window. Values are: •WindowDockStateDocked! •WindowDockStateFloating! •WindowDockStateTabbedDocument! •WindowDockStateTabbedWindow!
WindowState	WindowState (enumerated)	Specifies the state in which you want to run a window. Values are: •Maximized! •Minimized! •Normal! Do not change the WindowState property in the Open event of a window opened as a sheet.

Window property	Datatype	Description
WindowType	WindowType (enumerated)	Specifies the type of window. Values are: •Child! •Main! •MDI! •MDIHelp! •MDIDock! •MDIDockHelp! •Popup! •Response!
X	Integer	Specifies the X position (distance from left edge of screen) of the window, in PowerBuilder units. The values of the X coordinates in all windows except child windows are measured from the left side of the screen. In child windows, they are measured from the left side of the workspace of the parent window. The workspace is the area between the sides of the window (not including the thickness of the frame, toolbar, or scroll bar, if any) and the top and bottom of the window (not including the thickness of the border or the title bar, menu bar, toolbar, or scroll bar, if any). You cannot move a maximized window at runtime.
Y	Integer	Specifies the Y position (distance from the top of the screen) of the window, in PowerBuilder units. The values of the Y coordinates in all windows except child windows are measured from the top of the screen. In child windows, they are measured from the top of the workspace of the parent window. The workspace is the area between the sides of the window (not including the thickness of the frame, toolbar, or scroll bar, if any) and the top and bottom of the window (not including the thickness of the border or the title bar, menu bar, toolbar, or scroll bar, if any). You cannot move a maximized window at runtime.

Events

Scripts for events in a window and the controls in the window determine how the window behaves. Scripts control the action that is initiated when an event occurs within the window.

Window event	Occurs
Activate	Just before the window becomes active. When an Activate event occurs, the first object in the tab order for the window gets focus. If there are no enabled objects in the window, the window gets focus.
Clicked	When the user clicks in an unoccupied area of the window (any area with no visible, enabled object).
Close	When the window is closed.
CloseQuery	When you remove a window from display (close it). When you close a window, PowerBuilder triggers the CloseQuery event and then inspects the value of Message.ReturnValue. If the Message.ReturnValue is 1, the window cannot be closed. Closing any window causes PowerBuilder to close all child and pop-up windows that it opened, and closing an MDI Frame window causes PowerBuilder to close all sheet windows within it. Any window thus closed can set Message.ReturnValue to cancel the close operation.
Deactivate	When the window becomes inactive.
DoubleClicked	When the user double-clicks in an unoccupied area of the window (any area with no visible, enabled object).
DragDrop	When a dragged control is dropped on the window.
DragEnter	When a dragged control enters the window.
DragLeave	When a dragged control leaves the window.
DragWithin	When a dragged control is within the window.
Help	When the user presses the F1 key or drags the context help button (question mark) from the title bar to a menu item or control.
Hide	Just before the window is hidden.
HotLinkAlarm	After a Dynamic Data Exchange (DDE) server application has sent new (changed) data and the client DDE application has received it.
Key	When the user presses a key and the insertion point is not in a RichTextEdit or DataWindow edit control.
MouseDown	When the user presses the left mouse button in an unoccupied area of the window (any area with no visible, enabled object).
MouseMove	When the pointer is moved within the window.
MouseUp	When the user releases the left mouse button in an unoccupied area of the window (any area with no visible, enabled object).
Open	When a script executes the Open function for a window. The event occurs after the window has been opened but before it is displayed.
Other	When a Windows message occurs that is not a PowerBuilder event.
RButtonDown	When the right mouse button is pressed in an unoccupied area of the window (any area with no visible, enabled object).
RemoteExec	When a DDE client application has sent a command.

Window event	Occurs
RemoteHotLinkStart	When a DDE client application wants to start a hot link.
RemoteHotLinkStop	When a DDE client application wants to end a hot link.
RemoteRequest	When a DDE client application requests data.
RemoteSend	When a DDE client application has sent data.
Resize	When the user or a script opens or resizes a window.
Show	When a script executes the Show function for this window. The event occurs just before the window is displayed.
SystemKey	When the user presses Alt or Alt plus another key, except when the insertion point is in a DataWindow control or RichTextEdit control.
Timer	When a specified number of seconds elapses after the Timer function has been called.
ToolbarMoved	In an MDI frame window, when the user moves the FrameBar or SheetBar.

Functions

The following functions can trigger events in a window, manipulate or change a window, or provide information about a window.

PowerScript system functions
You can also use the PowerScript system functions in scripts for a window. For a list of the PowerScript system functions, see the Browser.

Opening and closing a window
Use the Open function to open a window and the Close function to close a window. Open and Close are system functions and are not listed here.

Window function	Datatype returned	Description
ArrangeSheets	Integer	Arranges the sheets or icons in the specified MDI frame window.
ChangeMenu	Integer	Changes the menu associated with a window.
ClassName	String	Returns the name assigned to the window.
CloseChannel	Integer	Closes a DDE channel.
CloseUserObject	Integer	Removes the specified user object from view, closes it, and executes its Destructor event.
CommitDocking	Integer	After all persisted sheets are opened, this function arranges them and makes them visible.

Window function	Datatype returned	Description
ExecRemote	Integer	Asks a DDE server application to execute the specified command.
GetActiveSheet	Window	Returns the currently active sheet in the specified MDI frame window.
GetCommandDDE	Integer	Obtains the command sent by the client application when your application is a DDE server.
GetCommandDDEOrigin	Integer	When called by the DDE server application, obtains the application name parameter used by the DDE client sending the command.
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service.
GetDataDDE	Integer	Obtains data sent from another DDE application and stores it in the specified string variable.
GetDataDDEOrigin	Integer	Determines the origin of data from a hot-linked DDE server application or a DDE client application and, if successful, stores the application's DDE identifiers in the specified strings.
GetFirstSheet	Window	Returns the top sheet in the MDI frame.
GetNextSheet	Window	Returns the sheet that is behind the specified sheet in the MDI frame.
GetParent	PowerObject	Returns a reference to the name of the parent object.
GetRemote	Integer	Asks a DDE server application to provide data and stores that data in the specified variable.
GetToolbar	Integer	Gets the values of the Visible, Alignment, and Title properties of the toolbar.
GetToolbarPos	Integer	Gets position information (coordinates) for a floating toolbar.
Hide	Integer	Makes the window invisible.
LoadDockingState	Integer	Loads two arrays of equal size: type names of persisted sheets and the corresponding IDs.
Move	Integer	Places the window in a new location specified by the X and Y arguments. The Move function does not move a maximized or minimized window.
OpenChannel	Long	Opens a channel to a DDE server application.
OpenSheet	Integer	Opens a sheet within an MDI (multiple document interface) frame window and creates a menu item for selecting the sheet on the specified menu.
OpenSheetAsDocument	Integer	Opens a sheet as a document within an MDI frame window for dockable windows.

Window function	Datatype returned	Description
OpenSheetDocked	Integer	Opens a sheet docked in a specified position within an MDI frame window for dockable windows.
OpenSheetFromDockingState	Integer	Opens one or more persisted sheets within an MDI frame window for dockable windows.
OpenSheetInTabGroup	Integer	Opens a sheet in a tab group within an MDI frame window for dockable windows.
OpenSheetWithParm	Integer	Opens a sheet within an MDI (multiple document interface) frame window and creates a menu item for selecting the sheet on the specified menu, as OpenSheet does and also stores a parameter in the system's Message object.
OpenSheetWithParmAsDocument	Integer	Opens a sheet as a document within an MDI frame window for dockable windows and stores a parameter in the system's Message object.
OpenSheetWithParmDocked	Integer	Opens a sheet docked in a specified position within an MDI frame window for dockable windows and stores a parameter in the system's Message object .
OpenSheetWithParmFromDockingState	Integer	Opens one or more persisted sheets within an MDI frame window for dockable windows and stores a parameter in the system's Message object.
OpenSheetWithParmInTabGroup	Integer	Opens a sheet in a tab group within an MDI frame window for dockable windows and stores a parameter in the system's Message object.
OpenUserObject	Integer	Displays the specified user object, making its properties available to scripts.
OpenUserObjectWithParm	Integer	Displays the specified user object, making its properties available to scripts, and stores a parameter in the system’s Message object.
ParentWindow	Window	Returns the parent window of the window.
PointerX	Integer	Returns the distance from the left edge of the screen to the pointer.
PointerY	Integer	Returns the distance from the top of the screen to the pointer.
PostEvent	Boolean	Adds an event to the end of the message queue for the window.
Print	Integer	Prints the window.
Resize	Integer	Changes the size of the window to the size specified in the width and height arguments. The Resize function does not resize a minimized or maximized window.
RespondRemote	Integer	Sends a DDE message indicating whether the command or data received from a remote DDE application was acceptable.
SaveDockingState	Integer	Stores the MDI state in the registry.

Window function	Datatype returned	Description
SetDataDDE	Integer	Sends data to a DDE client application when PowerBuilder is acting as a DDE server.
SetFocus	Integer	Sets focus to the specified window.
SetMicroHelp	Integer	Sets the MicroHelp text in the specified MDI frame window.
SetPosition	Integer	Specifies the position of the window in the front-to-back order of the application.
SetRedraw	Integer	Turns on or off automatic redrawing of the window after every change.
SetRemote	Integer	Asks a DDE server application to accept data and store it in the specified location.
SetSheetID	Integer	Sets the unique identifier for an open sheet.
SetToolbar	Integer	Sets the values of the Visible, Alignment, and Title properties of the toolbar.
SetToolbarPos	Integer	Sets the position of a fixed toolbar.
Show	Integer	Makes the window visible.
StartHotLink	Integer	Establishes a hot link with a DDE server application so that PowerBuilder is notified immediately of any changes in the specified data.
StartServerDDE	Integer	Establishes your application as a DDE server.
StopHotLink	Integer	Terminates a hot link with a DDE server application.
StopServerDDE	Integer	Causes your application to stop acting as a DDE server application.
TriggerEvent	Integer	Sends an event to a window control and executes the script associated with the event.
TypeOf	Object	Returns the type of the window.
WorkSpaceHeight	Integer	Returns the height of the workspace of the specified window. The workspace is the area between the sides of the window (not including the thickness of the frame, toolbar, or scroll bar, if any) and the top and bottom of the window (not including the thickness of the border or the title bar, menu bar, toolbar, or scroll bar, if any).
WorkSpaceWidth	Integer	Returns the width of the workspace of the specified window.
WorkSpaceX	Integer	Returns the distance from the left edge of the screen to the left edge of the workspace of the specified window.
WorkSpaceY	Integer	Returns the distance from the top of the screen to the top of the workspace of the specified window.

WSConnection object

The WSConnection object lets you set user-related, session-related, and authentication information when you connect to and process data from a Web service data source.

Properties

WSConnection property	Datatype	Description
AuthenticationMode	String	Specifies the authentication mode you want to use. This can be "basic" or "digest". These AuthenticationMode values are described on the Microsoft MSDN Web site at http://msdn.microsoft.com/en-us/library/aa833874(VS.80).aspx.
ClassDefinition	PowerObject	An object of type PowerObject containing information about the class definition of the object or control.
ClientCertificateFile	String	The name of the certificate file or files you want to use to connect to a Web service. The string value could include local files with a full path and URLs to remote certificate files. You must use a semicolon as a separator for multiple files.
Endpoint	String	Specifies a URL for the remote Web Service and tells the Web Service engine where the Web service resides. If the endpoint is not set, the Web Service engine uses the default endpoint embedded in the WSDL file.
Password	String	Specifies the password of the user who will consume the Web service.
ProxyServerHostName	String	Specifies the name of the proxy server host if the client machine is behind a firewall.
ProxyServerPassword	String	Specifies a password for the current user if a proxy server requests a user name and password. If the client machine is directly connected to the Internet, this property does not need to be set.
ProxyServerPort	UInteger	Specifies the port number of a proxy server if the client machine is behind a firewall.
ProxyServerUserName	String	Specifies the user name for a proxy server if the client machine is behind a firewall. If the client machine is directly connected to the Internet, this property does not need to be set.
Timeout	Long	Specifies the timeout period in seconds when invoking the Web service. The default value is 0, which does not set a timeout period for the Web service connection on the client side. (The Web service might still have a timeout value on the server side.)

WSConnection property	Datatype	Description
UserDomain	String	Specifies the domain the user is working in. This property is used together with UserName and Password properties for a fully qualified user identification.
UserName	String	Specifies the name of the user who will consume the Web service.
UseWindowsIntegrated Authentication	Boolean	Specifies whether the connection object uses integrated Windows authentication. Values are: TRUE – The application uses Windows authentication. FALSE – The application does not use Windows authentication. If this option is set to true, you do not need to set the UserName, Password, or UserDomain properties.

Events

WSConnection event	Occurs
Constructor	Immediately before the Open event occurs in the window
Destructor	Immediately after the Close event occurs in the window

Functions

WSConnection function	Datatype returned	Description
ClassName	String	Returns the name assigned to the control
GetContextService	Integer	Creates a reference to a context-specific instance of the specified service
GetParent	PowerObject	Returns a reference to the name of the parent object
PostEvent	Boolean	Adds an event to the end of the message queue for control
TriggerEvent	Integer	Triggers a specified event in the control and executes the script for the event
TypeOf	Object	Returns the type of the control

CHAPTER 3

Property Descriptions and Usage

About this chapter

This chapter lists the properties for PowerBuilder controls. For properties specific to controls in DataWindow objects, see the *DataWindow Reference*.

Contents

The properties are listed alphabetically.

Accelerator

Applies to

Controls that accept user input, including list boxes, MultiLineEdit, SingleLineEdit, ListView, EditMask, and TreeView

Description

Accelerator keys allow users to select an item (that is, change focus) with a keystroke rather than the mouse. An underlined character in the item's name or label tells the user what key to press. The user presses it in combination with the Alt key. If the currently selected control is not an editable control (such as a SingleLineEdit, MultiLineEdit, ListBox, or DropDownListBox), you need only press the accelerator key.

Accelerator keys are different from shortcut keys, which are defined key combinations that provide a quick way to accomplish certain tasks.

PowerBuilder term	Windows term
accelerator key	mnemonic access character
shortcut key	shortcut key or accelerator key

Usage

In a painter

❖ **To select a character as an accelerator key**

- Type the character into the Accelerator box on the General page of the control's Properties view.

  For example, to set m as the accelerator, type `m` in the box.

Accelerators for unlabeled controls To show the user what accelerator key to use for an unlabeled control or box, define StaticText to act as a label. Include an ampersand (&) before the character you want underlined. For example, in the StaticText control's General page, set the Text property to a value like `Edit &Maintenance Data` for a drop-down list that has `m` as an accelerator key. If you want an ampersand to display in the text, type two ampersands, and if you want an ampersand to display and serve as the accelerator key, type three ampersands.

In scripts

The Accelerator property is an integer consisting of the ASCII value of the accelerator key. Both of the following lines set m as the accelerator character for a MultiLineEdit control:

```
mle_1.Accelerator=77
mle_1.Accelerator = ASC("M")
```

AccessibleDescription

Applies to

Windows and controls that inherit from DragObject

Description

A description of the control and/or its purpose for use by accessibility tools such as readers for visually impaired users. You do not need to supply a description if the AccessibleName and AccessibleRole properties adequately describe the control, as in the case of a button with the label OK. You should provide a description for a picture or report control.

Usage

In a painter

On the Other page in the Properties view, type a description in the AccessibleDescription text box.

In a script

The Accessible Description property takes a string value. The following statement sets the AccessibleDescription property for a command button in a Window:

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

AccessibleName

Applies to

Windows and controls that inherit from DragObject

Description

A label that briefly describes the control, such as the text in a button or the name of a menu item, for use by accessibility tools such as readers for visually impaired users.

Usage

In a painter

On the Other page in the Properties view, type a name in the AccessibleName text box.

In a script

The AccessibleName property takes a string value. The following statement sets the AccessibleName property for a command button in a Window:

```
cb_1.accessiblename = "Delete"
```

AccessibleRole

Applies to

Windows and controls that inherit from DragObject

Description

A description of the kind of user interface element that the control is, for use by accessibility tools such as readers for visually impaired users. The description is a member of the AccessibleRole enumerated variable. The default role is defaultrole! and is used when the role cannot be determined. The following table lists the appropriate settings for PowerBuilder controls.

Table 3-1: AccessibleRole values for PowerBuilder controls

Control	AccessibleRole
Animation	animationrole!
CheckBox	checkbuttonrole!
CommandButton	pushbuttonrole!
DataWindow	clientrole!
DropDownListBox	comboboxrole!
DropDownPictureListBox	comboboxrole!
EditMask	textrole!
Graph	diagramrole!
GroupBox	groupingrole!
HProgressBar, VProgressBar	progressbarrole!
HScrollBar, VScrollBar	scrollbarrole!
HTrackBar, VTrackBar	sliderrole!
ListBox	listrole!
ListView	listrole!
MonthCalendar	clientrole!
MultiLineEdit	textrole!
Picture	graphicrole!
PictureButton	pushbuttonrole!
PictureHyperLink	linkrole!
PictureListBox	listrole!
RadioButton	radiobuttonrole!
RichTextEdit	clientrole!
SingleLineEdit	textrole!
StaticHyperLink	linkrole!
StaticText	statictextrole!
Tab	clientrole!
TabPage	clientrole!
TreeView	outlinerole!

Usage

In a painter

On the Other page in the Properties view, select a role from the AccessibleRole drop-down list.

In a script

The AccessibleRole property takes a value of the AccessibleRole enumerated variable. The following statement sets the AccessibleRole property for a command button in a Window:

```
cb_1.accessiblerole = pushbuttonrole!
```

Activation

Applies to

OLE controls

Description

Specifies how the user activates the control. Choices are:

- **Double Click** When the user double-clicks on the control, the server application is activated.
- **Get Focus** When the user clicks or tabs to the control, the server is activated. If you also write a script for the GetFocus event, do not call MessageBox or any function that results in a change in focus.
- **Manual** The control can be activated programmatically only with the Activate function.

During development, you activate the object in the Window painter.

Usage

In a painter

❖ **To specify how the object is activated:**

- Select the desired setting from the Activation drop-down list on the General page of the control's Properties view.

  The control can always be activated programmatically, regardless of the Activation setting.

In scripts

The Activation property takes a value of the omActivation enumerated datatype.

This example changes the Activation property type to ActivateOnGetFocus! for the object ole_1:

```
ole_1.Activation = ActivateOnGetFocus!
```

AdditionalOpts

Applies to

MLSynchronization and MLSync objects

Description

Specifies a command line option or a list of command line options for the dbmlsync synchronization command.

For information about available command line options, you can click the Usage button next to the Additional Options text box on the MobiLink Client Additional Options page of the MobiLink wizard, or you could open the chapter on synchronization parameters in the *MobiLink Clients* book.

Usage

In a painter

❖ **To specify additional command line options**

- On the Settings page of the object's Properties view, type the options you want in the Additional Options text box.

In scripts

You can include a string with multiple options to be added to a synchronization call.

For example, the following line sets log file verbosity for all messages except connection information and the MobiLink password (-v+), allows you to run in a minimized window (-q), and closes the window on completion of synchronization (-k option):

```
mySync_1.AdditionalOpts = '-v+ -q -k'
```

Alignment

Applies to

Controls that display text

Description

For most controls that display text, the alignment property specifies the alignment of all the text in the control. Text can be centered, left aligned, or right aligned.

For RichTextEdit controls, each paragraph has its own alignment setting, including Center, Left, Right, and Justify.

Usage

In a painter

❖ **To specify text alignment for controls other than RichTextEdit:**

- On the General page of the control's Properties view, select an alignment from the Alignment drop-down list, or click the Left, Center, and Right alignment buttons in the StyleBar. Use the StyleBar to set the alignment for several selected objects at once.

❖ **To align text in paragraphs in RichTextEdit controls:**

1 Select text in the paragraphs to be aligned.

2 Right-click on the selection to display the text object's property sheet, and select the alignment setting.

❖ **To allow alignment at runtime (RichTextEdit only):**

- On the Document tab page of the control's Properties view, check either the Toolbar or the PopMenu check box.

  At runtime, users select the text to align with the mouse and then right-click on the text to invoke a pop-up menu, or select an alignment button on the StyleBar.

In scripts

The datatype of the Alignment property is the Alignment enumerated datatype. It has four values: Center!, Left!, and Right! apply to all controls with text; Justify! applies only to RichTextEdit controls.

For example, the following line specifies center alignment for a MultiLineEdit control:

```
mle_1.Alignment = Center!
```

AllowEdit

Applies to

DropDownListBox, DropDownPictureListBox, and DatePicker controls

Description

When AllowEdit is enabled in a drop-down list, the user can edit the selection in the text box of the control. If AllowEdit is not enabled, the user can only make a selection from the list and cannot edit the selection.

In a DatePicker control, the user can modify the date in the control by picking a date from the drop-down calendar or by modifying the selected part of the date (year, month, or day) in the control if the format of the part is numeric. When AllowEdit is enabled, pressing F2 or placing the cursor in the control selects all the text in the control for editing. When the control loses focus, the text in the control reverts to its original value and the UserString event is fired. In the UserString event script, you can parse the string entered by the user and change the value in the control if the string passes validation.

To change the date by modifying one part of the date at a time when AllowEdit is enabled, the user can tab into the control and use the arrow keys to move between parts of the date.

When a part of the date is changed, the change is retained if it is valid. If the user enters a year, month, or day that is out of the range specified for the control, that value reverts to its previous value. If the user enters a day that is greater than the number of days in the month in the control, or a month greater than 12, the second digit entered is retained in the control.

Usage

In a painter

❖ **To allow editing:**

- On the General page of the control's Properties view, select the AllowEdit check box.

In scripts

The AllowEdit property takes a boolean value.

This example sets AllowEdit for a DropDownListBox:

```
ddlb_1.AllowEdit = TRUE
```

AnimationName

Applies to

Animation controls

Description

Specifies the name of the AVI file that contains the animation to be associated with the Animation control. The AVI file must be an uncompressed file or a file compressed using run-length encoding (BI_RLE8). You cannot use an AVI file that has a sound channel; it will not display in the control.

Usage

In a painter

❖ **To specify an AVI file for the control:**

- Click the browse (...) button next to the AnimationName field to select a file.

In scripts

The AutoPlay property takes a string value. The following line sets the AnimationName property for a control called am_1:

```
am_1.AnimationName = "C:\work\avifiles\Search.AVI"
```

AnimationTime

Applies to

Window controls

Description

Specifies the time in milliseconds that an opening or closing animation effect associated with a window takes to complete.

Usage

Use the AnimationTime property to control the number of milliseconds an opening or closing animation takes to execute. The value must be a positive integer in the range 1 to 5000 milliseconds. The default value is 200 milliseconds. While an opening or closing animation executes, the application waits for it to complete, so in general you should keep the animation time short.

In a painter

❖ **To set the AnimationTime property on a window:**

- Select or type a value in the AnimationTime spin control on the General page of the window's Properties view.

In scripts

The following statement sets the AnimationTime property for the w_splash window to 500 milliseconds:

```
w_splash.AnimationTime = 300
```

See also

CloseAnimation
OpenAnimation

AuthenticateParms

Applies to

MLSync and SyncParm objects

Description

Specifies a comma-separated list of authentication parameters for the remote database connection to the MobiLink synchronization server. If you set the AuthenticateParms property on an MLSync object, or if you call SetParm after setting the AuthenticateParms property on a SyncParm object, PowerBuilder inserts a -ap option with the AuthenticateParms value in the MLSync object's subsequent Synchronize call.

Usage

At design time, you can enter AuthenticateParms values on the General tab of the Properties view for an MLSync object. At runtime, application users can enter AuthenticateParms values on the Subscriptions tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can modify AuthenticateParms values in PowerScript directly on the synchronization object or indirectly, by setting it on the SyncParms object before calling SetParm.

```
SyncParms Parms
MLSync mySync
long rc
mySync = CREATE MLSync
Parms.MLUser = '50'
Parms.MLPass = 'xyz123'
Parms.AuthenticateParams = 'param1, param2, param3'
mySync.SetParm(Parms)
rc = mySync.Synchronize()
```

AutoArrange

Applies to

ListView controls

Description

When AutoArrange is enabled, PowerBuilder arranges icons automatically in large and small icon views.

Usage

In a painter

❖ **To enable automatic icon arrangement:**

- Select the AutoArrange check box on the General page of the ListView control's Properties view.

In scripts

The AutoArrange property takes a boolean value. The following line specifies automatic arrangement of icons in a ListView.

```
lv_1.AutoArrange = TRUE
```

AutoHScroll

Applies to

DropDownListBox, DropDownPictureListBox, EditMask, MultiLineEdit, SingleLineEdit controls

Description

When automatic horizontal scrolling is enabled, text in the control scrolls left or right automatically when the user enters or deletes data. When automatic horizontal scrolling is not enabled, text does not scroll left or right as the user changes data, and data that exceeds the width of the line provided is ignored.

Usage

In a painter

❖ **To enable automatic horizontal scrolling:**

- Select the AutoHScroll check box on the General page of the object's Properties view.

In scripts

The AutoHScroll property takes a boolean value.

For example, this statement enables automatic right and left scrolling as the user enters or modifies data in the edit box of a DropDownListBox control:

```
ddlb_1.AutoHScroll = TRUE
```

This property cannot be set at runtime for EditMask controls.

Automatic

Applies to

CheckBox and RadioButton controls

Description

When the Automatic property is enabled, the state of the control changes automatically when the user selects it. Typically, the state toggles between *selected* and *not selected*. For check boxes, if the ThreeState property has been enabled, the state of the control also toggles to a *third state*.

When this property is enabled, a mark is displayed in the control when the control's state is *selected* and no mark is displayed when the control's state is *not selected*. For check boxes, if the ThreeState property is enabled, a grayed out mark is displayed for the *third state*.

Usage

In a painter

❖ **To enable automatic state change:**

- Select the Automatic check box on the General page of the control's Properties view.

In scripts

The Automatic property takes a boolean value. This example sets a CheckBox so that its state changes each time it is selected.

```
cbx_1.Automatic = TRUE
```

AutoPlay

Applies to

Animation controls

Description

When the AutoPlay property is enabled, the AVI clip starts playing as soon as the animation control is opened.

Usage

In a painter

❖ **To start the animation automatically:**

- Select the Autoplay check box on the General page of the control's Properties view.

In scripts

The AutoPlay property takes a boolean value. The following line sets the AutoPlay property to true:

```
am_1.AutoPlay = TRUE
```

AutoScale

Applies to

grAxis objects in Graph controls

Description

AutoScale is a property of the Category and Value grAxis objects that are part of graph controls. There are three grAxis objects: Category, Series, and Values.

Autoscale specifies whether or not to scale the axis of the Graph automatically to the minimum and maximum values for the data.

Usage

In a painter

❖ **To enable autoscaling:**

1 Display the Axis tab page on the graph's Properties view.

2 Select the desired Axis from the Axis drop-down list.

3 Select the AutoScale check box, if it is enabled.

AutoScale is enabled only if it is applicable to the selected graph type and axis.

In scripts

The AutoScale property takes a boolean value.

The following line turns off autoscaling for the Values axis in the Graph gr_emp.

```
gr_emp.Values.AutoScale = FALSE
```

AutoSize

Applies to

MonthCalendar controls and grDispAttr objects within Graph controls

Description

For MonthCalendar controls, when the AutoSize property is on, the calendar is sized to hold a single month. If you change other properties that affect size, such as TextSize and TodaySection, the calendar resizes automatically both at design time and runtime.

For grDispAttr objects, the AutoSize property allows PowerBuilder to change the font size of the text object automatically according to the amount of text being displayed. If automatic sizing is not enabled, you must set the text size.

Usage

In a painter

❖ **To enable automatic sizing of calendar controls:**

- Select the AutoSize check box on the General page in the Properties view.

❖ **To enable automatic sizing of text objects in graphs:**

1. Display the Text tab page of the control's Properties view.
2. Select a text object from the Text Object list.
3. Select the AutoSize check box.

   If you clear the AutoSize check box for a specific text object, set the text size for that object by selecting a value from the TextSize list.

In scripts

The AutoSize property takes a boolean value.

For the MonthCalendar control, AutoSize is true by default. This example turns autosizing off for a MonthCalendar control:

```
mc_1.AutoSize = false
```

For grDispAttr objects, AutoSize can be set using the grDispAttr object for each text component.

This example turns autosizing off for the graph control's title and then sets a specific text size.

```
gr_1.TitleDispAttr.AutoSize = FALSE
gr_1.TitleDispAttr.TextSize = 10
```

This example turns autosizing off for the label of the Category Axis of the graph control and then sets a specific text size.

```
gr_1.Category.LabelDispAttr.AutoSize = FALSE
gr_1.Category.LabelDispAttr.TextSize = 8
```

AutoSkip

Applies to

EditMask controls

Description

When the AutoSkip property is enabled, the user's cursor automatically skips to the next control in the tabbing order after entering all the characters allowed by the mask. If AutoSkip is not enabled, the cursor does not skip automatically to the next control.

Usage

In a painter

❖ **To enable Auto Skip:**

- Select the AutoSkip check box on the Mask tab page of the EditMask control's Properties view.

In scripts

The AutoSkip property takes a boolean value. This example enables automatic skipping to the next control.

```
em_1.AutoSkip = TRUE
```

AutoVScroll

Applies to

EditMask, MultiLineEdit controls

Description

When automatic vertical scrolling is enabled, text in the control scrolls up or down automatically when the user enters or modifies data. When automatic vertical scrolling is not enabled, text does not scroll up or down automatically as the user changes data, and data that exceeds the height of the space provided is ignored.

Usage

In a painter

❖ **To enable automatic vertical scrolling:**

- Select the AutoVScroll check box on the General page of the object's Properties view.

In scripts

The AutoVScroll property takes a boolean value. The following example enables automatic vertical scrolling:

```
mle_1.AutoVscroll = TRUE
```

This property cannot be set at runtime for EditMask controls.

BackColor

Applies to

Windows and most controls

Description

The BackColor property defines the color to be used for the background of an object. When you are defining the background color in a painter, some of the choices take their values from the current Windows color scheme or from custom colors. To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view. You can also choose Transparent.

Usage

In a painter

❖ **To set the background color for a window, MonthCalendar control, or graph:**

- Select a color from the BackColor drop-down list on the General page in the window's or graph's Properties view.

❖ **To set the background color for other controls:**

- Select a color from the BackColor drop-down list on the Font tab page in the control's Properties view.

❖ **To set the background color for text objects in graphs:**

1 Select the desired text object in the Text Object list box on the Text tab page of the graph control's Properties view.

2 Select a color from the BackColor drop-down list.

In scripts

The BackColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color of windows and other objects. The BackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the background color for a graph control:

```
gr_1.BackColor = RGB(255, 255, 0)
```

For text displayed in the graph, BackColor is a property of a text component, such as labels on an axis.

The following example sets background color to blue for text labels on the Category axis of the graph control:

```
gr_1.Category.LabelDispAttr.BackColor = RGB(0, 128,
255)
```

BeginX

Applies to

Line controls

Description

The BeginX property specifies the X position in PowerBuilder units of the beginning of the line.

The X coordinate is the distance from the left edge of the window or custom user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To specify the beginning X coordinate of the line:**

- Insert the line. If you want to change the beginning location, change the value of the BeginX field on the Position tab page of the line's Properties view.

In scripts

The BeginX property takes an integer value. The following example sets the beginning of the line at 1000 PowerBuilder units from the left edge of the window or user object and 500 PowerBuilder units from the top edge:

```
ln_1.BeginX = 1000
ln_1.BeginY = 500
```

BeginY

Applies to

Line controls

Description

The BeginY property specifies the Y position in PowerBuilder units of the beginning of the line.

The Y coordinate is the distance from the top edge of the window or custom user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To specify the beginning Y coordinate of the line:**

- Insert the line. If you want to change the beginning location, change the value of the BeginY field on the Position tab page of the line's Properties view.

In scripts

The BeginY property takes an integer value. The following example sets the beginning of the line at 500 PowerBuilder units from the top edge of the window or user object and 1000 PowerBuilder units from the left edge.:

```
ln_1.BeginY = 500
ln_1.BeginX = 1000
```

BoldSelectedText

Applies to

Tab controls

Description

When the BoldSelectedText property is enabled, the label for a tab page becomes bold when the user selects the tab page. If this property is not enabled, the tab text of the selected tab page has the same appearance as the tab text of the other tab pages.

Usage

In a painter

❖ **To enable the BoldSelectedText property:**

- Select the BoldSelectedText check box on the General page of the Tab control's Properties view.

In scripts

The BoldSelectedText property takes a boolean value. The following example specifies that labels on tab pages of the tab_1 control are bold when they are selected:

```
tab_1.BoldSelectedText = TRUE
```

Border

Applies to

Windows, other controls

Description

When the Border property is enabled, the window or control has a border.

Some controls and window types always have borders, whether or not the Border property is enabled.

Usage

In a painter

❖ **To specify a border:**

- Select the Border check box on the General page of the window's or control's Properties view.

In scripts

The Border property takes a boolean value. The following example specifies that a static text control should display a border and sets the border style to 3D Lowered:

```
st_1.Border = TRUE
st_1.BorderStyle = StyleLowered!
```

BorderColor

Applies to

StaticText and StaticHyperLink controls

Description

BorderColor defines the color to be used for the border around a StaticText or StaticHyperLink control. The color is visible only with the Box border style.

To add your own colors to the BorderColor list, select Design>Custom Color before displaying the Properties view.

Usage

In a painter

❖ **To specify a border color:**

1 Select the Border check box on the General page of the control's Properties view.

2 Select StyleBox! from the BorderStyle list.

3 Select the desired color from the BorderColor list on the General page.

In scripts

The BorderColor property takes a long value. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

This example enables the display of a border, specifies the Box border style, and then specifies red as the border color for StaticText control st_1:

```
st_1.Border = TRUE
st_1.BorderStyle = StyleBox!
st_1.BorderColor = RGB(255,0,0)
```

BorderStyle

Applies to

Most controls

Description

The BorderStyle property lets you define the border appearance of a control. Styles include 3D Lowered, 3D Raised, Box, and Shadow Box.

Usage

In a painter

❖ **To set the border style:**

- Select the desired style from the BorderStyle list on the General page of the control's Properties view.

  To turn the border off, uncheck the Border check box on the General page.

In scripts

To change the appearance of the border, set Border to true and set the BorderStyle property to a value of the BorderStyle enumerated datatype.

The following example sets the border for a DropDownListBox.

```
ddlb_1.Border = TRUE
ddlb_1.BorderStyle = StyleLowered!
```

BottomMargin

Applies to

RichTextEdit controls

Description

The BottomMargin property specifies the size in inches of the bottom margin on the printed page.

Usage

In a painter

❖ **To set the bottom margin:**

- Enter the desired size in inches in the BottomMargin field of the Document tab page of the RichTextEdit control's Properties sheet.

In scripts

The BottomMargin property takes a long value. The following example sets the bottom margin of a printed page of a RichTextEdit control to 1 inch:

```
rte_1.BottomMargin = 1
```

BringToTop

Applies to

Windows and controls

Description

For windows, this property specifies whether PowerBuilder moves the window to the top of the front-to-back order. For windows, this property can be set only in scripts.

For controls, this property specifies whether PowerBuilder moves the control to the top of the front-to-back order within the window.

Usage

In a painter

❖ **To set BringToTop property for controls:**

1 Select the control.

2 Right-click on the control and select Bring to Front or Send to Back from the Pop-up menu, or select Format>Bring to Front or Format>Send to Back from the menu bar.

In scripts

BringToTop takes a boolean value.

This statement brings the window in front of other windows in the application:

```
w_1.BringToTop = TRUE
```

ButtonHeader

Applies to

ListView controls

Description

When the ButtonHeader property is enabled, the column titles in a ListView's report view appear as pushable buttons instead of labels.

Usage

In a painter

❖ **To set the ButtonHeader property:**

- Select the ButtonHeader check box on the General page of the ListView control's Properties view.

In scripts

The ButtonHeader property is only relevant to a report view in a ListView control. To enable report view, you must write a script that establishes columns with the AddColumn and SetColumn functions, and then populate the columns using the SetItem function.

See "Using Lists in a Window" in *Application Techniques* for more information about using report view.

The ButtonHeader property takes a boolean value. The following example specifies a button header for a report in a ListView:

```
lv_1.ButtonHeader = TRUE
```

Cancel

Applies to

CommandButton, PictureButton, OLECustomControl controls

Description

Cancel defines whether the Esc key activates the button or control. If the Cancel property is enabled, the Esc key triggers the control's Clicked event. If Cancel is not enabled, the control does not respond to the Esc key.

If you enable Cancel for more than one control, the last one set responds to the Esc key.

Usage

In a painter

❖ **To enable the Cancel property:**

- Select the Cancel check box on the General page of the control's Properties view.

In scripts

The Cancel property takes a boolean value. The following line allows the CommandButton to respond to the Esc key:

```
cb_1.Cancel = TRUE
```

CalendarBackColor

Applies to

DatePicker and EditMask controls

Description

The CalendarBackColor property defines the color to be used for the background of the calendar that displays when you click on the down arrow in a DatePicker control or an EditMask control with the DropDownCalendar property set to true. When you are defining the background color in a painter, some of the choices take their values from the current Windows color scheme or from custom colors. To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

❖ **To set the background color for the calendar in a DatePicker or EditMask control:**

- Select a color from the BackColor drop-down list on the Calendar page in the control's Properties view.

In scripts

The CalendarBackColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color . The CalendarBackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the background color for the calendar in a DatePicker control:

```
dp_1.CalendarBackColor = RGB(255, 255, 0)
```

CalendarTextColor

Applies to

DatePicker and EditMask controls

Description

The CalendarTextColor property specifies the color to be used for text in the calendar in a DatePicker control or an EditMask control with the DropDownCalendar property set to true.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

❖ **To set the text color for the calendar in a DatePicker or EditMask control:**

- Select a color from the TextColor drop-down list on the Calendar page in the control's Properties view.

In scripts

The CalendarTextColor property is a long indicating the color to be used for the text for an object. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following line sets the text color for the calendar in a DatePicker control to blue:

```
dp_1.CalendarTextColor = RGB(0,0,255)
```

CalendarTitleBackColor

Applies to

DatePicker and EditMask controls

Description

The CalendarTitleBackColor property defines the color to be used for the background of the calendar's title.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select a color from the CalendarTitleBackColor drop-down list on the Calendar page in the Properties view.

In scripts

The CalendarTitleBackColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color of the month or months in a calendar. The CalendarTitleBackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets pale green as the background color for titles:

```
dp_1.CalendarTitleBackColor = RGB(128, 255, 128)
```

CalendarTitleTextColor

Applies to

DatePicker and EditMask controls

Description

The CalendarTitleTextColor property specifies the color used for text in the calendar's title.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select the desired color from the TitleTextColor drop-down list on the Calendar tab page of the Properties view.

In scripts

The CalendarTitleTextColor property is a long indicating the color to be used for the title for a calendar in a DatePicker control or an EditMask control with the DropDownCalendar property set to true. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

For example, the following line sets the title text color for the control *dp_1*:

```
dp_1.CalendarTitleTextColor = RGB(0,0,255)
```

CalendarTrailingTextColor

Applies to

DatePicker and EditMask controls

Description

The CalendarTrailingTextColor property specifies the color used for text for leading and trailing days in the calendar.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select the desired color from the TrailingTextColor drop-down list on the Calendar tab page of the Properties view.

In scripts

The CalendarTrailingTextColor property is a long indicating the color to be used for leading and trailing days in the calendar. These are days in months that are partly displayed in the calendar. In a calendar showing a single month, they are the last few days of the preceding month and the next few days of the following month. In a calendar showing the three months July to September, the leading days are the last few days of June and the trailing days are the first few days of October. The default color is the color defined for Disabled Text.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following line sets the trailing text color for the calendar in a DatePicker control to “Inactive Title Bar”:

```
dp_1.CalendarTrailingTextColor = 134217731
```

Category

Applies to

Graph controls

Description

The Category property of the Graph control allows you to specify the properties of the category axis of the graph. The category axis is an object (of the type grAxis) within the Graph control.

Usage

In a painter

❖ **To set the properties of the Category Axis of a graph control:**

1 Display the Axis tab page of the graph control's Properties view.

2 Select Category from the Axis drop-down list.

3 Set the desired values on the Axis tab page.

In scripts

The datatype of the Category property is grAxis, which is a type of object that has its own properties for controlling the appearance of the axis. Use the following syntax to specify values for the category axis:

GraphControlName.Category.*grAxisProperty* = value

The following example sets the label of the category axis of a graph control:

```
gr_1.Category.Label = "Types of Products"
```

CategorySort

Applies to

Graph controls

Description

The CategorySort property specifies how the categories are sorted: ascending, descending, or unsorted.

Usage

In a painter

❖ **To specify how the categories are sorted:**

- Select the desired sort type from the CategorySort drop-down list on the General page of the graph control's Properties view.

In scripts

The CategorySort property takes a value of the grSortType enumerated datatype, which has the values Ascending!, Descending!, Unsorted!, and UserDefinedSort!.

The following example specifies that the categories should be unsorted:

```
gr_1.CategorySort = Unsorted!
```

Center

Applies to

Windows

Description

Causes the window to be centered on the screen when it is created or sized.

Usage

In a painter

❖ **To set the Center property for a window:**

- Select the Center check box on the General page of the window's Properties view.

In scripts

The Center property takes a boolean value. The following example sets the Center property for the window, w_mine:

```
w_mine.Center = TRUE
```

Checked

Applies to

CheckBox, RadioButton controls, and Menu objects

Description

When the value of the Checked property is TRUE, the item or control is selected.

Control	Displayed when selected
CheckBox	X displays in the check box
Menu item	Check mark displays next to the item
RadioButton	Center of button becomes dark

If Checked is false, the item or control is not selected.

Usage

In a painter

❖ **To set the Checked property on a control:**

- Select the Checked check box on the General page of the control's Properties view.

❖ **To set the Checked property on a menu item:**

1 Select the menu item in the Menu painter.

2 Select the Checked check box on the General page.

In scripts

The Checked property takes a boolean value. The following example sets the Checked property of a RadioButton to true, which causes the button to be selected.

```
rb_1.Checked = TRUE
```

For menu items in drop-down or cascading menus, you can also use the Check and Uncheck functions. See the descriptions of those functions for examples of equivalent syntax.

CloseAnimation

Applies to

Window controls

Description

Specifies an optional animation effect that displays when the window closes.

Usage

The CloseAnimation property takes a value of the WindowAnimationStyle enumerated variable. For "slide" values, the whole window appears to slide from the direction selected. For "roll" values, the window does not move but its display is cleared from the direction selected. Values are:

- NoAnimation! (default) – The window closes with no animation.
- TopSlide! – The window slides from the top to the bottom of its extent.
- BottomSlide! – The window slides from the bottom to the top of its extent.
- LeftSlide! – The window slides from the left to the right of its extent.
- RightSlide! – The window slides from the right to the left of its extent.
- TopRoll! – The window rolls from the top to the bottom of its extent.
- BottomRoll! – The window rolls from the bottom to the top of its extent.
- LeftRoll! – The window rolls from the left to the right of its extent.
- RightRoll! – The window rolls from the right to the left of its extent.
- FadeAnimation! – The window fades out.

- CenterAnimation! – The window collapses from the center.

You can modify the animation properties at any time and use them for any window type. They are most often used in pop-up windows.

FadeAnimation! can be used only in top-level windows. It does not work in child windows. In MDI applications, you cannot use FadeAnimation! for sheet windows. Fading affects the transparency of the window, and sheet windows in MDI applications always inherit the transparency of the frame window.

Also, if the Transparency property of a frame or main window is set to a value above 0, the TopRoll!, BottomRoll!, LeftRoll!, RightRoll!, and CenterAnimation! settings do not work with sheet windows or child windows. All settings work correctly for frame windows.

While the animation executes, the application waits for it to complete. Use the AnimationTime property to control the number of milliseconds the animation takes to execute.

The window's CloseQuery event is triggered before the animation begins. The Close event is triggered when the animation has completed.

In a painter

❖ **To set the CloseAnimation property on a window:**

- Select a value from the CloseAnimation drop-down list on the General page of the window's Properties view.

In scripts

The following example sets the CloseAnimation property of the w_splash window to FadeAnimation!:

```
w_splash.CloseAnimation = FadeAnimation!
```

See also

AnimationTime
OpenAnimation

CollectionMode

Applies to

InkPicture controls

Description

Specifies whether ink only, gestures only, or ink and gestures are collected.

Usage

In a painter

❖ **To specify the mode of ink collection:**

- Select a value from the CollectionMode drop-down list on the Ink page in the Properties view.

In scripts

The CollectionMode property takes a value of the InkCollectionMode enumerated variable. Values are:

- InkOnly! – Only ink is collected, creating a stroke. The Gesture event is not triggered.
- InkAndGesture! – Ink and single-stroke gestures are collected (the default).
- GestureOnly! – Only gestures are collected. Gestures can be single or multiple strokes. CollectionMode must be set to GestureOnly! for the double-tap gesture to be recognized.

This example specifies that the InkPicture control will recognize gestures but not collect ink:

```
ie_1.InkCollectionMode = GestureOnly!
```

ColumnsPerPage

Applies to

Windows and user objects

Description

The ColumnsPerPage property determines the number of columns on a page for scrolling purposes. The default is 0 (10 columns per page). PowerBuilder multiplies UnitsPerColumn by ColumnsPerPage to determine the number of PowerBuilder units to scroll the window horizontally when the user clicks in the scroll bar.

For information on calculating ColumnsPerPage and UnitsPerColumn, see Scrolling in windows and user objects on page 591.

Usage note
To control the vertical scroll bar in a window or user object, use the UnitsPerLine and LinesPerPage properties.

Usage

In a painter

❖ **To set the ColumnsPerPage property:**

- Enter the desired number (between 1 and 100) in the ColumnsPerPage option on the Scroll tab page of the window's Properties view.

In scripts

The ColumnsPerPage property takes an integer value between 1 and 100. The following line sets ColumnsPerPage for a window to 20:

```
This.ColumnsPerPage = 20
```

ContentsAllowed

Applies to

OLE controls

Description

Specifies whether the OLE object in the control must be embedded or linked or whether either type of OLE object can be inserted at runtime.

Usage

In a painter

❖ **To set the ContentsAllowed property:**

- Select the desired option from the Contents drop-down list on the General page of the control's property page.

  Choices are Any, Embedded, or Linked.

In scripts

The datatype of the ContentsAllowed property is omContentsAllowed. The following example sets the value of the ContentsAllowed property to linked:

```
ole_1.ContentsAllowed = ContainsLinkedOnly!
```

ControlCharsVisible

Applies to

RichTextEdit controls

Description

Specifies whether control characters (carriage returns, spaces, and tabs) are visible. This property can be enabled and disabled by the user at runtime from the toolbar and from the Properties item of the pop-up menu, if the PopMenu property is enabled.

Usage

In a painter

❖ **To make control characters visible:**

- Select the ControlCharsVisible check box on the Document tab page in the Properties view of the RichTextEdit control.

In scripts

The ControlCharsVisible property takes a boolean value. The following line enables display of control characters in a RichTextEdit control:

```
rte_1.ControlCharsVisible = TRUE
```

ControlMenu

Applies to

Windows and DataWindow controls

Description

The Control Menu property specifies whether the Control Menu box appears in the title bar of the Window or DataWindow control.

Usage

In a painter

❖ **To display the Control Menu box:**

- Select the ControlMenu check box on the General page of the window's or DataWindow control's Properties view.

In scripts

The ControlMenu property takes a boolean value. The following example specifies that the Control Menu box will appear in the title bar of the DataWindow control dw_1.

```
dw_1.ControlMenu = TRUE
```

CornerHeight

Applies to

RoundRectangle controls

Description

The Corner Height property sets the radius of the vertical part of the corners of a RoundRectangle control. The radius is in PowerBuilder units.

Usage

In a painter

❖ **To set the CornerHeight property:**

- Enter an integer in the CornerHeight field on the General page of the control's Properties view.

In scripts

The CornerHeight property takes an integer. This example sets the corner height for a RoundRectangle rr_1:

```
rr_1.CornerHeight = 10
```

CornerWidth

Applies to

RoundRectangle controls

Description

The Corner Width property sets the radius of the horizontal part of the corners of a RoundRectangle control. The radius is in PowerBuilder units.

Usage

In a painter

❖ **To set the CornerWidth property:**

- Enter an integer in the CornerWidth field on the General page of the control's Properties view.

In scripts

The CornerWidth property takes an integer. This example sets the corner width for a RoundRectangle rr_1:

```
rr_1.CornerWidth = 10
```

CreateOnDemand

Applies to

Tab controls

Description

When CreateOnDemand is enabled, all controls on all tab pages of a Tab control are instantiated when the Tab control is created, but the Constructor event of controls on hidden tab pages is not triggered until the user views the tab page either by clicking on the tab page or by calling the SelectTab function (the SelectTab function sets the SelectedTab property). Until the Constructor event for a control has run, a graphical representation of the control is not created, and its handle is not available. Constructor events for controls on the selected tab page are always triggered when the Tab control is created.

A window opens more quickly if the creation of a graphical representation is delayed for tab pages with many controls. However, scripts cannot refer to a control on a tab page until its Constructor event has run and a graphical representation has been created.

Usage

In a painter

❖ **To set the CreateOnDemand property:**

- Select the CreateOnDemand check box on the General page of the tab control's Properties view.

In scripts

The CreateOnDemand property takes a boolean value. The following example specifies that graphical representations of tab pages are not created until the tab page is selected:

```
tab_1.CreateOnDemand = TRUE
```

For more information about using this property in scripts, see the chapter on using tab controls in a window in *Application Techniques*.

CustomFormat

Applies to

DatePicker controls

Description

The CustomFormat property specifies a custom format for the display of the date in a DatePicker control. You must set the Format property to the enumerated value dtfCustom! for the CustomFormat to take effect.

The format strings in the following table can be combined to format the date and time. If you want to include string literals in the CustomFormat string, enclose them in single quotes to ensure that the letters they contain are not interpreted as format strings.

Format string	Description
d	The one- or two-digit day.
dd	The two-digit day. Single-digit day values are preceded by 0.
ddd	The three-character weekday abbreviation.
dddd	The full weekday name.
h	The one- or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single-digit values are preceded by 0.
H	The one- or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single-digit values are preceded by 0.
m	The one- or two-digit minute.
mm	The two-digit minute. Single-digit values are preceded by 0.
M	The one- or two-digit month.
MM	The two-digit month. Single-digit values are preceded by 0.
MMM	The three-character month abbreviation.
MMMM	The full month name.
s	The one- or two-digit second.
ss	The two-digit second. Single-digit values are preceded by 0.
t	The one-letter abbreviation for AM or PM in uppercase (AM displays as A).
tt	AM or PM in uppercase.
X	A callback field used to query the user for a portion of the custom format. This format string is not supported in PowerBuilder.
y	The one-digit year (2007 displays as "7").
yy	The two-digit year (2007 displays as "07").
yyy *or* yyyy	The full year (2007 displays as "2007").

Usage

In the painter

❖ **To set the CustomFormat property:**

1 Select dtfcustom! from the Format drop-down list on the General page of the control's Properties view.

2 Specify a custom format in the Custom Format field.

This custom format displays the date and time at 8 p.m. on the first day of 2007 as `January 1, 2007 08:00:00 PM`:

```
MMMM d, yyyy hh:mm:ss tt
```

This custom format displays the same time on the last day of 2006 as December 31, 2007 20:00:00 PM:

```
MMMM d, yyyy HH:mm:ss
```

In scripts

The CustomFormat property takes a string value. The following statements set the Format property to allow a custom format, then set the custom format to display the full month name, the two-digit date followed by a comma, and the full year:

```
dp_1.Format = dtfCustom!
dp_1.CustomFormat = "MMMM dd, yyyy"
```

DataObject

Applies to

DataWindow controls and DataStore objects

Description

The DataObject property specifies the name of the DataWindow object or Report object associated with the DataWindow control or DataStore.

Usage

In a painter

❖ **To specify the name of the DataWindow object contained in a DataWindow control:**

- Enter the name of an existing DataWindow object in the DataObject field on the General property page of the DataWindow control's Properties view, or use the Browse button to select an object.

In scripts

The DataObject property takes a string. This example specifies d_employ as the DataWindow object in the DataWindow control dw_1.

```
dw_1.DataObject = "d_employ"
```

Reinitializing the DataWindow control or DataStore

Setting the DataObject to an empty string reinitializes the DataWindow control or DataStore and removes all storage associated with the previous DataObject.

DataSource

Applies to

MLSync objects

Description

Specifies the ODBC data source name used to connect to a SQL Anywhere remote database. Setting this property is equivalent to using the `-c "dsn=myDSN"` dbmlsync option, where *myDSN* is the data source name.

Usage

This is a required property of the synchronization object. If the property value is not set before a synchronize call, the synchronization fails and an error string is saved to the synchronization object's ErrorText property.

At design time, you can enter a DataSource property value on the SQL Anywhere Connect tab of the Properties view for an MLSync object. At runtime, application users can enter DataSource values in the DSN text box on the SQL Anywhere tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can modify DataSource values in script as follows:

```
mlSync.DataSource = "myDSN"
```

DataType

Applies to

grAxis objects in Graph controls

Description

DataType is a property of the grAxis objects that can be part of graph controls. There are three grAxis objects: Category, Series, and Values.

DataType specifies the type of data that is assigned to the axis. Possible datatypes are Number, Date, DateTime, Default, Double, Text, and Time.

Usage

In a painter

❖ **To specify datatype:**

1 Display the Axis tab page of the graph control's Properties view.

2 Select the desired Axis type from the Axis drop-down list.

3 Select the desired datatype from the DataType drop-down list.

In scripts

The DataType property takes a value of type grAxisDataType.

The following line sets the DataType of the Values axis of a graph:

```
gr_1.Values.DataType = AdtDate!
```

DBPass

Applies to

MLSync and SyncParm objects

Description

Password for the SQL Anywhere remote database. Setting this property value is equivalent to including the `-c "pwd=myPassword"` dbmlsync option, where *myPassword* is the password for the database connection.

Usage

At design time, you can enter a DBPass value on the SQL Anywhere Connect tab of the Properties view for an MLSync object. At runtime, application users can enter DBPass values on the SQL Anywhere tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can modify DBPass values in script as follows:

```
mySync.DBPass = "myPassword"
```

DBUser

Applies to

MLSync and SyncParm objects

Description

User ID for the SQL Anywhere remote database. Setting this property value is equivalent to including the `-c "uid=myUserID"` dbmlsync option, where *myUserID* is the user name for the database connection.

Usage

At design time, you can enter a DBUser value on the SQL Anywhere Connect tab of the Properties view for an MLSync object. At runtime, application users can enter DBUser values on the SQL Anywhere tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can modify DBUser values in script as follows:

```
mySync.DBUser = "myUserID"
```

Default

Applies to

CommandButton, PictureButton, OLECustomControl controls

Description

The Default property specifies that the control is the default button. If Default is true, the selected control has a thick border and receives a Clicked event when the user presses Enter (unless the user has tabbed to another control). If Default is false, the control is not the default and pressing Enter does not affect it unless the user tabs to it.

Setting focus

If the window contains an editable field, such as a MultiLineEdit, then the default button behaves as expected (receives the Clicked event when the user presses Enter) when focus is on the editable field. When the user presses Tab to move focus to another button (not the default), pressing Enter fires the Clicked event for the button that currently has focus.

If the window does not contain an editable field, use SetFocus or tab order to make sure the default button behaves as documented above.

You can make a CommandButton, PictureButton, or OLECustomControl control the default button so that it responds to the Enter key. If you check Default for more than one control, the last one set acts as the default.

Usage

In a painter

❖ **To enable the Default property**

- Select the Default check box on the General page of the control's Properties view.

In scripts

The Default property takes a boolean value. To set a PictureButton as the default button, use a line like the following:

```
pb_1.Default = TRUE
```

DeleteItems

Applies to

ListView, TreeView controls

Description

When the DeleteItems property is true, the user can delete items from the ListView or TreeView with the Delete key. When DeleteItems is false, the user cannot delete items.

Usage

In a painter

❖ **To allow users to delete items from the control:**

- Select the DeleteItems check box on the General page of the control's Properties view.

In scripts

The DeleteItems property takes a boolean value. The following example disables deletion of ListView items by the user:

```
lv_1.DeleteItems = FALSE
```

Depth

Applies to

Graph controls

Description

For 3-D graphs, specifies the depth of the graph as a percentage of its width. The default is 100 percent.

Usage

In a painter

❖ **To set the Depth property:**

1 Display the General page of the graph control's Properties view.

2 Select a 3-D type of graph from the GraphType drop-down list.

3 Use the Depth slider control to set the Depth to the desired percentage of width.

In scripts

The Depth property takes an integer. This example specifies that the depth of the graph is 50% of its width:

```
gr_1.Depth = 50
```

DisabledName

Applies to

PictureButton controls

Description

The DisabledName property specifies the name of a picture file to be displayed when the PictureButton is disabled. The picture can be in the following formats:

- bitmap (*.BMP*)
- runlength encoded (*.RLE*)
- Windows metafile (*.WMF*)
- GIF (*.GIF*)
- JPEG (*.JPG* or *.JPEG*)

Usage

In a painter

❖ **To specify a picture to be displayed when the button is disabled:**

- Enter the name of the file in the DisabledName field on the General page of the control's Properties view, or use the Browse button next to the DisabledName field to select a file.

In scripts

The DisabledName property takes a string containing the name of a file. The string can include the path. This example specifies the picture *controls.bmp* for the disabled view of the PictureButton:

```
pb_1.DisabledName = "d:\pbhelp\controls.bmp"
```

DisableDragDrop

Applies to

TreeView controls

Description

The DisableDragDrop property determines whether events for dragging, such as BeginDrag, are triggered when the user clicks on an item within the control and drags. DisableDragDrop affects only the dragging of items within the control.

When DisableDragDrop is true, no drag events occur when the user tries to drag an item. To implement drag and drop, write scripts for the appropriate dragging events.

DisableDragDrop also affects when selection occurs. When it is true, an item the user clicks is selected when the mouse button is pressed down. When it is false, the item is selected when the mouse button is released.

Usage

In a painter

❖ **To disable drag and drop within the TreeView control:**

- Select the DisableDragDrop check box on the General page of the control's Properties view.

In scripts

The DisableDragDrop property takes a boolean value. The following example prevents drag events from being triggered within a TreeView control:

```
tv_1.DisableDragDrop = TRUE
```

DisableNoScroll

Applies to

ListBox and PictureListBox controls

Description

The DisableNoScroll property specifies the behavior of a scroll bar in a list box. If the property is enabled, the scroll bar is always visible, but it is disabled when all the items can be accessed without it. If the property is disabled, the scroll bar is displayed only if it is necessary, based on the number of items and the height of the ListBox or PictureListBox.

Usage

In a painter

❖ **To make the scroll bar always visible but disabled when not needed:**

- Select the DisableNoScroll check box on the General page of the control's Properties view.

In scripts

The DisableNoScroll property takes a boolean value. This example for a ListBox displays the scroll bar only when needed:

```
lb_1.DisableNoScroll = FALSE
```

DisplayEveryNLabels

Applies to

grAxis objects of Graph controls

Description

DisplayEveryNLabels is a property of the grAxis objects that can be part of graph controls. There are three grAxis objects: Category, Series, and Values.

This property specifies which major divisions to label on the selected axis in the graph. For example, a value of 2 means to label every other tick mark. Use 0 to let the graph select the optimum number of labels to use.

Usage

In a painter

❖ **To specify the number of major divisions to label:**

1 Display the Axis tab page in the graph's Properties view.

2 Select the desired Axis from the Axis drop-down list.

3 Use the spin control to select a number from 0 to 100 in the DisplayEveryNLables field.

In scripts

The DisplayEveryNLabels property takes an integer. The following example sets labeling at every 10 tick marks for the Series Axis:

```
gr_1.Series.DisplayEveryNLabels = 10
```

DisplayExpression

Applies to

grDispAttr objects within Graph controls

Description

The DisplayExpression property specifies an expression whose value is the label for a specified text object within the Graph control.

The default expression is the value of the property containing the text for the graph component.

Usage

In a painter

❖ **To specify a display expression for a text object:**

1 Display the Text tab page of the Graph control's Properties view.

2 Select the text object for which you want to define a display expression from the Text Object list box.

The default value of the DisplayExpression property is displayed in the DisplayExpression field.

3 Specify the display expression in the Display Expression field, or click the More button to display the Modify Expression dialog box.

In scripts

The DisplayExpression property can be set using the grDispAttr object for each text component. DisplayExpression takes a string, which can contain an expression.

The following example appends today's date to the title of the graph:

```
gr_1.TitleDispAttr.DisplayExpression = 'title + " " + Today()'
```

DisplayName

Applies to

Application object, OLE controls

Description

DisplayName is a user-readable name for your application or OLE control. This name is displayed in OLE dialog boxes and windows that show the object's name. If you do not specify a value, the name of the control (such as ole_1) or application (value of the AppName property) is used for Display Name.

Usage

In a painter

❖ **To set a DisplayName for an OLE control:**

- Enter the name in the DisplayName field of the General page of the control's Properties view.

❖ **To set a DisplayName for an application:**

1 Open the application in the Application painter.

2 Enter the name in the DisplayName field of the General page of the application's Properties view.

In scripts

The DisplayName property takes a string. The following example sets a name for an OLE control:

```
ole_1.DisplayName = 'My Project'
```

DisplayOnly

Applies to

MultiLineEdit, SingleLineEdit, RichTextEdit, EditMask controls

Description

When the Display Only property is enabled, users cannot change the text in an editable control. If the property is not enabled, users can change the text.

Usage

In a painter

❖ **To specify that text is display only:**

- Select the Display Only check box on the General page of the control's Properties view.

In scripts

The DisplayOnly property takes a boolean value. The following example specifies that text in a MultiLineEdit control cannot be changed:

```
mle_1.DisplayOnly = TRUE
```

DisplayType

Applies to

OLE controls

Description

The DisplayType property specifies how the OLE object is displayed in the control. The control can display the actual contents, an icon to represent the object, or as an ActiveX document. ActiveX documents fill the space of the object container and have access to all features of the server application.

Usage

In a painter

❖ **To set the display type of the control:**

- Select the desired value from the Display Type drop-down list on the General page of the control's Properties view.

In scripts

The DisplayType property takes a value of the omDisplayType enumerated datatype. Values are:

- DisplayAsContent
- DisplayAsActiveXDocument!
- DisplayAsIcon!

The following example sets the DisplayType to icon:

```
ole_1.DisplayType = DisplayAsIcon!
```

DocumentName

Applies to

RichTextEdit controls

Description

The Document Name property specifies the name that appears in the print queue when the user prints the contents of the control.

Usage

In a painter

❖ **To set the document name for printing:**

- Enter the document name in the Document Name for Printing field on the Document tab page of the control's Properties view.

In scripts

The DocumentName property takes a string. The following example specifies a document name for the print queue for a RichTextEdit control:

```
rte_1.DocumentName = "Report 1"
```

DragAuto

Applies to

Draggable controls

Description

The DragAuto property determines whether PowerBuilder puts the control into drag mode automatically. If the property is enabled, when the user clicks the control and starts dragging it, PowerBuilder puts the control in drag mode. Clicking the control triggers a DragDrop event, not a Clicked event.

If DragAuto is not enabled, then when the user clicks the control, PowerBuilder does not put the control in drag mode. You have to call the Drag function to put the control into drag mode.

Usage

In a painter

❖ **To set DragAuto:**

- Select the DragAuto check box on the Other tab page of the control's Properties view.

In scripts

Most controls have a DragAuto property. It takes a boolean value. The following example sets drag mode for a CommandButton.

```
cb_1.DragAuto = TRUE
```

DragIcon

Applies to

Draggable controls

Description

The DragIcon property specifies the icon to display when the user drags the control. The default icon is a box the size of the control.

When the user drags a control, the icon displays when the cursor is over an area in which the user can drop the control (a valid drop area). When the cursor is over an area that is not a valid drop area, the no-drop icon displays.

Usage

In a painter

❖ **To specify the drag icon:**

1 Display the Other tab page of the control's Properties view.

2 Click the down arrow on the DragIcon field and select a stock icon from the list of stock icons, or use the Browse button to select another icon (*.ICO*) file.

In scripts

The DragIcon property takes a string containing the name of the icon file you want to display when the user drags the control. You can specify a stock icon or any icon filename.

The following example sets the drag icon for a ListBox to an icon called *arrow.ico*:

```
lb_1.DragIcon = 'c:\examples\arrow.ico'
```

This example sets the drag icon to the stock icon Question:

```
lb_1.DragIcon = 'Question!'
```

DropDownCalendar

Applies to

EditMask controls

Description

Specifies that the control uses a drop-down calendar to display and select dates when the MaskDataType is DateMask! or DateTimeMask!.

Usage

In a painter

❖ **To set the DropDownCalendar property:**

- Select or clear the DropDownCalendar check box on the Mask page in the Properties view

In scripts

The DropDownCalendar property takes a boolean value. The default is false. This example specifies that the EditMask control uses a drop-down calendar:

```
em_1.DropDownCalendar = true
```

DropDownRight

Applies to

DatePicker and EditMask controls

Description

Specifies whether the drop-down calendar is aligned with the right or left side of the DatePicker or EditMask control.

Usage

In a painter

❖ **To set the DropDownRight property:**

- Select or clear the DropDownRight check box on the General page in the Properties view for DatePicker controls or the Calendar page for EditMask controls

In scripts

The DropDownRight property takes a boolean value. The default is false (the calendar is left aligned). This example specifies that the calendar is aligned with the right side of the DatePicker control:

```
dp_1.DropDownRight = true
```

DropLines

Applies to

grAxis objects in Graph controls

Description

A drop line is a line that extends from a data point to its axis. Drop lines are not available for all graph types.

Usage

In a painter

❖ **To set the drop line type:**

1 Display the Axis tab page of the graph control's Properties view.

2 Select the desired Axis from the Axis drop-down list.

3 Select the type of line desired from the DropLines drop-down list.

In scripts

The DropLines property takes a value of the LineStyle enumerated datatype.

The following example sets dashed lines for the drop lines in the Series axis:

```
gr_1.Series.DropLines = Dash!
```

EditLabels

Applies to

ListView and TreeView controls

Description

When EditLabels is enabled, the user can edit labels in the ListView or TreeView by selecting the item, clicking on the label, and then adding or deleting characters. When EditLabels is not enabled, the labels are not editable.

Usage

In a painter

❖ **To enable editing of labels:**

- Select the Edit Labels check box on the General page of the control's Properties view.

In scripts

The EditLabels property takes a boolean value. The following example enables editing of labels in a TreeView:

```
tv_1.EditLabels = TRUE
```

EditMode

Applies to

InkPicture controls

Description

Specifies whether the editing mode of the control is set for drawing ink, editing ink, or deleting ink.

Usage

In a painter

❖ **To specify the mode of ink collection:**

- Select a value from the EditMode drop-down list on the Ink page in the Properties view.

In scripts

The EditMode property takes a value of the InkPicEditMode enumerated variable. Values are:

- InkPicDeleteMode! – Ink is deleted.
- InkPicInkMode! – Ink can be drawn (default).
- InkPicSelectMode! – Ink is selected for editing.

This example specifies that the InkPicture control delete any ink under the tip of the stylus:

```
ie_1.EditMode = InkPicDeleteMode!
```

Elevation

Applies to

Graph controls

Description

Elevation determines how much of the full perspective of a 3D graph is visible. It specifies the angle of front-to-back elevation.

Elevation is disabled for 2D graphs.

Usage

In a painter

❖ **To change the elevation of a 3-D graph:**

- Move the Elevation slider on the General page of the graph's Properties view.

In scripts

The Elevation property takes an integer value. The following example specifies an elevation of 35:

```
gr_1.Elevation = 35
```

Enabled

Applies to

All graphic controls except drawing objects and progress, scroll, and track bars.

Description

When the Enabled property is enabled, the control can have focus. Users can select the control by clicking on it. If the control is included in the tab order, users can tab to it.

If the Enabled property is not enabled, the control cannot have focus and the user cannot select it.

For a MonthCalendar control, the enabled property enables or disables keyboard input.

Enabled does not affect whether the control is visible (see Visible).

Usage

In a painter

❖ **To set the Enabled property:**

- Select the Enabled check box on the General page of the control's Properties view.

In scripts

The Enabled property takes a boolean value. Most controls have an Enabled property. This example sets Enabled for a CommandButton:

```
cb_1.Enabled = TRUE
```

EncryptionKey

Applies to

MLSynchronization, MLSync, and SyncParm objects

Description

Specifies an encryption key for SQL Anywhere remote database. Setting this property is equivalent to using the `-c "dbkey=myKey"` dbmlsync option, where *myKey* is the encryption key for the database.

Usage

At design time, you can enter an encryption key value on the SQL Anywhere Connect tab of the Properties view for an MLSync object. At runtime, application users can enter encryption key values on the SQL Anywhere tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can modify the EncryptionKey values in script as follows:

```
mySync_1.EncryptionKey = "myKey"
```

EndX

Applies to

Line controls

Description

The EndX property specifies the X coordinate of the end of the line in PowerBuilder units.

The X coordinate is the distance from the left edge of the window or custom user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To set the X coordinate of the end of the line:**

- Insert the line. If you want to change the ending location, change the value of the EndX field on the Position tab page of the line's Property view.

In scripts

The EndX property takes an integer value. This example sets the X coordinate of the end of the line:

```
ln_1.EndX = 1200
```

EndY

Applies to

Line controls

Description

The EndY property specifies the Y coordinate of the end of the line in PowerBuilder units.

The Y coordinate is the distance from the top edge of the window or custom user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To set the Y coordinate of the end of the line:**

- Insert the line. If you want to change the ending location, change the value of the EndY field on the Position tab page of the line's Property view.

In scripts

The EndY property takes an integer value. This example sets the Y coordinate of the end of the line:

```
ln_1.EndY = 1200
```

ErrorText

Applies to

MLSynchronization, MLSync objects

Description

Write-only property used to store error and diagnostic messages generated when a synchronization function is called incorrectly.

Usage

The ErrorText value is returned to the synchronization object from the MobiLink Server.

In scripts

In the Clicked event of a command button, you can cause the ErrorText value to display in the multiline edit box of a status window as follows:

```
parent.mle_error.text = mlsync.errortext
```

Escapement

Applies to

grDispAttr objects in a graph control

Description

This property specifies the rotation for the baseline of the Axis text objects in a graph control.

Usage

In a painter

❖ **To set the rotation of text objects within a graph:**

1 Display the Text tab of the graph's Properties view.

2 Select the desired text object from the Text Object list box.

3 Use the Escapement spin control to set the desired value.

In scripts

Escapement takes an integer value that specifies the rotation in tenths of a degree. 0 is horizontal. A value of 900 rotates the text 90 degrees; 450 rotates the text 45 degrees. The following example sets the rotation of the Value Axis Label to 90 degrees:

```
gr_1.Value.LabelDispAttr.Escapement = 900
```

ExtendedOpts

Applies to

MLSynchronization and MLSync objects

Description

Specifies a command line option or a list of command line options for the dbmlsync synchronization command.

For information about available command line options, you can click the Usage button next to the Extended Options text box on the MobiLink Client Additional Options page of the MobiLink wizard, or you can open the chapter on synchronization parameters in the *MobiLink Clients* book.

Usage

In a painter

On the Settings page of the object's Properties view, type the options you want in the Extended Options text box.

In scripts

You can include a string with extended options to be added to a synchronization call.

For example, the following line sets the script version to "test":

```
mySync_1.ExtendedOpts = "sv = test"
```

On the next Synchronize call from the MLSync object mySync_1, PowerBuilder adds the -e extended option with the value that you set:

```
dbmlsync -e "sv=test"
```

ExtendedSelect

Applies to

ListBox, PictureListBox, ListView controls

Description

ExtendedSelect specifies whether users can select more than one item in a ListBox or ListView at one time. When ExtendedSelect is enabled, users can select multiple items by clicking on an item and dragging the mouse up or down to select items, using Click or Shift+Click to select a sequential group of items, or using Control+Click on multiple items. When ExtendedSelect is not enabled, users cannot select multiple items.

Usage note
If both MultiSelect and ExtendedSelect are enabled, then the behavior of ExtendedSelect takes precedence.

Usage

In a painter

❖ **To enable extended select:**

- Select the Extended Select check box on the General page of the control's property page.

In scripts

The ExtendedSelect property takes a boolean value. The following example lets the user select multiple items using extended selection techniques for a ListBox lb_1:

```
lb_1.ExtendedSelect = TRUE
```

FaceName

Applies to

Controls that can display text

Description

The FaceName property specifies the typeface used for text in the control. For tab controls, the property specifies the typeface for text labels on tabs.

The typefaces available for your use are those installed on your system. Keep in mind that the fonts available to you may not be available where you deploy your application.

Usage

This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.

In a painter

❖ **To set the typeface of text in a control:**

- Select a typeface from the FaceName list box on the Font tab page of the control's Properties view.

❖ **To set the typeface of text objects in a graph control:**

1. Display the Text tab page of the graph control's Properties view.
2. Select the desired text object from the Text Object list box.
3. Select a typeface from the FaceName list box.

In scripts

The FaceName property takes a string value. The following example sets the font for text labels on tab pages of a tab control to the Arial typeface:

```
tab_1.FaceName = "Arial"
```

Factoid

Applies to

InkEdit controls

Description

Specifies a context for ink recognition in an InkEdit control. Set this property if the input data is of a known datatype, such as a date, to constrain the search for a recognition result. Possible values include digit, email, Web, date, time, number, currency, percent, and telephone.

Usage

In a painter

❖ **To specify a factoid for an InkEdit control:**

- Select a factoid from the drop-down list.

The following values are available. After the Default and None factoids, the drop-down list displays factoids for special formats in alphabetical order, followed by single character factoids and Asian language factoids.

Factoid	Description
Default	Returns recognizer to the default setting. For Western languages, the default setting includes the user and system dictionaries, various punctuation marks, and the Web and Number factoids. For Eastern languages, the default setting includes all characters supported by the recognizer.
None	Disables all factoids, dictionaries, and the language model.
Currency	Currency in pounds, dollars, euros, and yen.
Date	Dates written in English; for example 8/19/2005, Aug 19, 2005, or Friday, August 19, 2005.
E-mail	Email addresses.
Filename	Windows file name paths. The name cannot include the following characters: / : " < > |
Number	Numeric values, including ordinals, decimals, separators, common suffixes, and mathematical symbols. This factoid includes the Currency and Time factoids.
Percent	A number followed by the percent symbol.
Postal Code	Postal codes as written in English, for example 01730 or CT17 9PW.
System Dictionary	Words in the system dictionary only.
Telephone	Telephone numbers as written in English, for example (555) 555 5555 or +44 1234 123456.
Time	Times as written in English, for example 15:05 or 3:05 pm.
Web	Various URL formats.
Word List	Words on the word list associated with the recognizer context only.
Digit	A single digit (0-9).
One Char	A single ANSI character.
Upper Char	A single uppercase character.

In addition, the following Asian language factoids are available:

Bopomofo	Kanji Common
Hangul Common	Katakana
Hiragana	Korean Common
Jamo	Simplified Chinese Common
Japanese Common	Traditional Chinese Common

In scripts

The Factoid property takes a string value.

This example sets the Factoid property to Telephone for the control ie_1:

```
ie_1.Factoid = Telephone
```

FillColor

Applies to

Oval, Rectangle, RoundRectangle controls

Description

The FillColor property defines the color used to fill the control. When you are defining the background color in a painter, some of the choices take their values from the current Windows color scheme or from custom colors.

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

Usage

In a painter

❖ **To set the fill color:**

- Select the desired color from the Fill Color drop-down list on the General page of the control's Properties view.

In scripts

The FillColor property takes a long value (-2 to 16,777,215) that specifies the numerical value of the background color of windows and other objects. The FillColor value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

This example specifies yellow as the fill color for the RoundRectangle rr_display:

```
rr_display.FillColor=RGB(255,255,0)
```

FillPattern

Applies to

Oval, Rectangle, RoundRectangle, StaticText, and StaticHyperLink controls

Description

The FillPattern property specifies the hatch pattern used to fill the control.

For drawing objects, the pattern uses the FillColor for the background and the LineColor for the foreground lines.

For StaticText and StaticHyperLink controls, the pattern uses the BackColor for the background and the TextColor for the foreground lines. The text and the pattern lines use the same color.

Usage

In a painter

❖ **To set the fill pattern:**

- Select the desired hatch pattern from the Fill Pattern drop-down list on the General page of the control's Properties view.

In scripts

The FillPattern property takes a value of the FillPattern enumerated datatype.

The following example sets a diamond fill for a StaticText control:

```
st_1.FillPattern = Diamond!
```

FirstDayOfWeek

Applies to

DatePicker, MonthCalendar controls

Description

Specifies which day of the week displays on the left in the calendar.

Usage

In a painter

❖ **To set the FirstDayOfWeek property:**

- Select a day from the FirstDayOfWeek drop-down list on the Calendar page in the Properties view for DatePicker controls or the General page in the Properties view for MonthCalendar controls.

In scripts

The FirstDayOfWeek property takes a value of the enumerated variable WeekDay. This example sets Monday as the first day of the week for a MonthCalendar control:

```
mc_1.FirstDayOfWeek = Monday!
```

FixedLocations

Applies to

ListView controls

Description

When the FixedLocations property is enabled, the user cannot drag items to new positions in the control. When Fixed Locations is not enabled and DragAuto is enabled, the user can drag items to new positions.

Usage

In a painter

❖ **To set the FixedLocation property:**

- Select the Fixed Locations check box on the General page of the control's Properties view

In scripts

The FixedLocations property takes a boolean value. The following example enables dragging of items within a ListView.

```
lv_1.DragAuto = TRUE
lv_1.FixedLocations = FALSE
```

FixedWidth

Applies to

Tab controls

Description

When the FixedWidth property is enabled, tabs have a fixed width. The width is determined by the longest tab label. When FixedWidth is not enabled, tabs shrink to the length of their text labels.

Usage

In a painter

❖ **To set the FixedWidth property:**

- Select the Fixed Width check box on the General page of the tab control's Properties view.

In scripts

The FixedWidth property takes a boolean value. The following example specifies that tabs in the control tab_1 have a fixed width:

```
tab_1.FixedWidth = TRUE
```

FocusOnButtonDown

Applies to

Tab controls

Description

When the FocusOnButtonDown property is enabled, each tab page gets focus when the user clicks on it. A dotted rectangle marks the tab page. If FocusOnButtonDown is not enabled, the clicked tab page does not display the focus rectangle. In either case, the selected tab page comes to the front.

Usage

In a painter

❖ **To set the FocusOnButtonDown property:**

- Select the Focus On Button Down check box on the General page of the tab control's Properties view.

In scripts

The FocusOnButtonDown property takes a boolean value. The following example specifies that tab pages within the control tab_1 display the focus rectangle when clicked:

```
tab_1.FocusOnButtonDown = TRUE
```

FocusRectangle

Applies to

Graph, Picture, PictureHyperLink, OLE, OLE Custom Control, StaticText, and StaticHyperLink controls

Description

When the FocusRectangle property is enabled, a dotted rectangle (the focus rectangle) displays when the control has focus. If this property is not enabled, the focus rectangle does not appear.

Usage

In a painter

❖ **To set the FocusRectangle property:**

- Select the Focus Rectangle check box on the General page of the control's Properties view.

In scripts

The FocusRectangle property takes a boolean value. The following example specifies that a focus rectangle will appear when the StaticText control has focus:

```
st_1.FocusRectangle = TRUE
```

FontCharSet

Applies to

Controls that can display text

Description

This property specifies the font character set to be used for the text in the control. Character sets and font typefaces are related, so choosing the wrong character set can cause a different font to be used than the one expected.

When working in a painter, setting the font face name property causes the correct character set to be selected.

Usage

In a painter

❖ **To set the font character set:**

- Select a font character set from the FontCharSet list box on the Font tab page of the control's Properties view.

❖ **To set the font character set of text objects in a graph control:**

1. Display the Text tab page of the graph control's Properties view.
2. Select the desired text object from the Text Object list box.
3. Select a font character set from the FontCharSet list box.

In scripts

The FontCharSet takes a value of the FontCharSet enumerated datatype. The following example sets the character set for a static text control to ANSI:

```
st_1.FontCharSet = ANSI!
```

FontFamily

Applies to

Controls that can display text

Description

The FontFamily property sets the type style used for the text in the control.

When working in a painter, setting the font face name causes the correct font family to be selected.

Usage

In a painter

❖ **To set the font type style:**

- Select a font style from the FontFamily list box on the Font tab page of the control's Properties view.

❖ **To set the font type style of text objects in a graph control:**

1 Display the Text tab page of the graph control's Properties view.

2 Select the desired text object from the Text Object list box.

3 Select a font style from the FontFamily list box.

In scripts

The FontFamily property takes a value of the FontFamily enumerated datatype. The following example sets the FontFamily for a static text control to Roman:

```
st_1.FontFamily = Roman!
```

FontPitch

Applies to

Controls that can display text

Description

The FontPitch property specifies the spacing of the font used for the text in the control.

When working in a painter, setting the font face name causes the correct font pitch to be selected.

Usage

In a painter

❖ **To set the font spacing:**

- Select a font spacing from the FontPitch list box on the Font tab page of the control's Properties view.

❖ **To set the font spacing of text objects in a graph control:**

1 Display the Text tab page of the graph control's Properties view.

2 Select the desired text object from the Text Object list box.

3 Select a font spacing from the FontPitch list box.

In scripts

The FontPitch property takes a value of the FontPitch enumerated datatype. The following example sets the font pitch for a static text control:

```
st_1.FontPitch = Fixed!
```

FontWeight

Applies to

DatePicker controls

Description

The FontWeight property specifies the stroke weight of the text in the control.

Usage

In the painter

❖ **To set the stroke weight of all text in a control:**

- Display the Font page in the control's Properties view and select the Bold check box, or select the control and click the B button on the StyleBar.

In scripts

The FontWeight property takes an integer value. A value of 400 indicates a normal weight, and 700 indicates a bold weight. The following example sets the text labels of the tab pages of a tab control to bold:

```
dp_1.FontWeight = 700
```

Format

Applies to

DatePicker controls and grDispAttr objects in a graph control

Description

DatePicker controls The Format property for DatePicker controls specifies the format of the date displayed in the DatePicker control. The property takes a value of the DateTimeFormat enumerated variable. Long and short date and time formats are determined by the regional settings in the Windows control panel on the local computer. Values are:

DtfCustom! – use the format specified in the CustomFormat property
DtfLongDate! – display a long date (default)
DtfShortDate! – display a short date
DtfTime!– display a time

grDispAttr objects The Format property for grDispAttr objects allows you to define display formats for text objects in graphs. Display formats are masks in which certain characters have special significance.

The characters you use for formatting depend on the datatype of the data. PowerBuilder supports four kinds of display formats:

- Numbers
- Strings
- Dates

- Times

You can specify colors in any display format by specifying a color keyword before the format.

For more information about using colors and each kind of display format, see Using colors with display formats on page 585 and the sections that follow it. For more information about defining display formats, see the PowerBuilder *Users Guide*.

Usage

In a painter

❖ **To set the display format for a DatePicker control:**

- Select a value from the Format drop-down list on the General page in the Properties view.

❖ **To set the display format for a text object:**

1. Display the Text tab page of the graph control's Properties view.
2. Select a text object in the Text Object list box.
3. Enter an expression in the DisplayExpression field or select a format from the Format drop-down list.

In scripts

The following line specifies that the DatePicker control should use the short date format:

```
dp_1.Format = dtfShortDate!
```

The CustomFormat property takes a string value. The following statements set the Format property to allow a custom format, then set the custom format to display the full month name and the two-digit date followed by a comma, and the full year:

```
dp_1.Format = dtfCustom!
dp_1.CustomFormat = "MMMM dd, yyyy"
```

For grDispAttr objects, each type of display format uses special characters that have special meaning for that format. The Format property takes a string value composed of these special characters.

The following example specifies a format for numeric data that always displays three digits, with two decimal places:

```
gr__1.Values.DispAttr.Format = "0.00"
```

The following example specifies a string format for alphanumeric data:

```
gr_1.Category.dispAttr.Format = "@@@/AAA"
```

Frame

Applies to

grAxis objects in Graph controls

Description

The Frame property specifies the line style used in the frame for an axis of a 3D Graph. The frame is the side of the 3D box associated with the selected axis.

Usage

In a painter

❖ **To set the Frame line style of an Axis:**

1 Select a 3D graph style on the General page of the graph control's Properties view. Not all 3D graph styles support the Frame property.

2 Display the Axis tab page of the Properties view and select the desired axis from the Axis drop-down list.

3 Select the desired line style from the Frame drop-down list in the Line Style group.

In scripts

The Frame property takes a value of the LineStyle enumerated datatype.

This example specifies a dashed line for the Series axis frame of Graph gr_1:

```
gr_1.Series.Frame = Dash!
```

FreeDBLibraries

Applies to

Application object

Description

Determines whether PowerBuilder libraries are held in memory after PowerBuilder disconnects from a database.

Usage

Prior to PowerBuilder 8, PowerBuilder automatically freed database interface libraries when it disconnected from the database. To enhance performance and resolve process initialization issues with certain database management systems, PowerBuilder no longer frees the database interface libraries by default when it disconnects. The FreeDBLibraries property on the Application object enables you to force the release of these libraries upon disconnecting from the database.

This is a runtime property only. To free libraries held in memory after PowerBuilder disconnects from a database at design time, select the Free Database Driver Libraries On Disconnect check box on the General page of the System Options dialog box. Design-time and runtime libraries are always cleared from memory on shutdown of PowerBuilder.

For more information on the design-time selection for freeing database interface libraries, see *Connecting to Your Database*.

In the application painter

❖ **To free PowerBuilder libraries upon disconnecting from a database:**

- Select the FreeDBLibraries check box on the General page of the application's Properties view.

In scripts

The FreeDBLibraries property takes a boolean value. The following example sets the property to clear memory and release PowerBuilder libraries after disconnecting from a database at runtime:

```
my_app.FreeDBLibraries = TRUE
```

GraphType

Applies to

Graph controls

Description

The GraphType property specifies the kind of graph: Area, Bar, Column, Line, Pie, or Scatter.

Usage

In a painter

❖ **To select the graph type:**

- Select the type of graph desired from the Graph Type drop-down list on the General page of the graph's Properties view.

  The graph displayed in the control changes to show an example of the selected type.

  The type of graph you select affects what properties are available on other tabs.

In scripts

The GraphType property takes a value of the grGraphType enumerated datatype. The following example defines the Graph gr_1 as a 3D pie chart:

```
gr_1.GraphType=Pie3D!
```

HasButtons

Applies to

TreeView controls

Description

When HasButtons is enabled, PowerBuilder displays + and - buttons next to parent items. The buttons indicate whether an item is expanded (-) or collapsed (+).

Usage

In a painter

❖ **To enable the display of buttons:**

- Select the HasButtons check box on the General page of the control's Properties view.

In scripts

The HasButtons property takes a boolean value. The following line specifies that PowerBuilder will display + and - buttons in a TreeView:

```
tv_1.HasButtons = TRUE
```

HasLines

Applies to

TreeView controls

Description

When the HasLines property is enabled, PowerBuilder connects tree items by lines.

Usage

In a painter

❖ **To enable connecting TreeView items:**

- Select the HasLines check box on the General page of the control's Properties view.

In scripts

The HasLines property takes a boolean value. The following line specifies that PowerBuilder will display lines connecting tree items:

```
tv_1.HasLines = TRUE
```

HeaderFooter

Applies to

RichTextEdit controls

Description

The HeaderFooter property specifies whether the control has a header/footer section. You must write a menu or button script to allow users to display the header and footer editing panels.

See "Implementing Rich Text" in *Application Techniques* for more information about using RichTextEdit controls.

Caution
If the RichTextEdit control does not have a header/footer section and you open a document that has a header and footer section, the header and footer is ignored. If you later save the document from within the control using the same file name, the header and footer in the original document are lost.

Usage

In a painter

❖ **To enable the header/footer section in the control:**

- Select the Header-Footer check box on the Document tab page of the control's Properties view.

In scripts

The HeaderFooter property takes a boolean value, but it can be set only in the control's Properties view. The value cannot be changed during execution.

Use the ShowHeadFoot function to display the header and footer editing panels during runtime.

Height

Applies to

Visible controls, windows

Description

The Height property specifies the height of a control or window in PowerBuilder units.

Usage

In a painter

❖ **To set the height of a control or window**

- Enter the desired height in the Height edit box on the Other tab page of the object's Properties view, or select the control or window and resize it with your cursor.

In scripts

The Height property takes an integer value specifying the height of an object in PowerBuilder units. The following example sets the height of a DataWindow control dw_1:

```
dw_1.Height = 750
```

It is illegal to resize a minimized or maximized sheet or frame. Changing the Width or Height property for a minimized or maximized window is not supported.

HideSelection

Applies to

SingleLineEdit, MultiLineEdit, EditMask, ListView, TreeView controls

Description

If the HideSelection property is enabled, selected text does not stay selected (highlighted) when the control does not have focus. If this property is not enabled, selected text stays highlighted when the control loses focus.

Usage

In a painter

❖ **To enable HideSelection:**

- Select the Hide Selection check box on the General page of the control's Properties view.

In scripts

The HideSelection property takes a boolean value. The following example specifies that selected text in a SingleLineEdit is always highlighted.

```
sle_1.HideSelection = FALSE
```

Host

Applies to

MLSynchronization and MLSync objects

Description

Specifies the machine name for the MobiLink synchronization server.

Usage

At design time, you can enter a value for Host on the MLServer tab of the Properties view for an MLSync object. At runtime, application users can enter a value for the Host machine on the MLServer tab page of the default synchronization options window generated by the MobiLink wizard.

If the host name is defined by subscriptions in the remote database, you do not need to set this property.

In scripts

You can change the Host name in script as follows:

```
mySync_1.Host = "myMachineName"
```

HScrollBar

Applies to

DataWindow, DropDownListBox, DropDownPictureListBox, EditMask, InkEdit, ListBox, PictureListBox, MultiLineEdit, and RichTextEdit controls, user objects, and windows

Description

When the HScroll Bar property is enabled, a horizontal scroll bar appears when all of the data cannot be displayed at one time. If this property is not enabled, no horizontal scroll bar appears.

Usage

In a painter

❖ **To set a horizontal scroll bar for controls:**

- Select the HScroll Bar check box on the General page of the control's Properties view.

❖ **To set a horizontal scroll bar for windows or user objects:**

- Select the HScroll Bar check box on the Scroll tab page of the window's or object's Properties view.

In scripts

The HScrollBar property takes a boolean value. The following example allows a horizontal scroll bar to appear when needed in a ListBox.

```
lb_1.HScrollBar = TRUE
```

This property cannot be set at runtime for EditMask controls.

HSplitScroll

Applies to

DataWindow controls

Description

If the HSplit Scroll property is enabled, the user can split the DataWindow control into two panes with separate scroll bars. The user moves the split bar to divide the DataWindow control into two panes.

If this property is not enabled, the user cannot split the DataWindow control.

Usage

In a painter

❖ **To allow splitting the control into two panes:**

- Select the HSplit Scrolling check box on the General page of the control's Properties view.

In scripts

The HSplitScroll property takes a boolean value. The following example allows splitting of a DataWindow control dw_1:

```
dw_1.HSplitScroll = TRUE
```

HTextAlign

Applies to

PictureButton controls

Description

The HTextAlign property specifies whether text in the PictureButton control is right aligned, left aligned, or centered horizontally.

Usage

In a painter

❖ **To set the horizontal alignment of text:**

- Select the desired alignment from the Horizontal Alignment drop-down list on the General tab of the control's Properties view, or use the Left, Right, and Center alignment buttons on the StyleBar.

In scripts

The HTextAlign property takes a value of the Alignment enumerated datatype.

The following example specifies right alignment for text in a PictureButton.

```
pb_1.HTextAlign = Right!
```

Icon

Applies to

DataWindow controls and windows

Description

The Icon property specifies the icon to display when the DataWindow control or window is minimized. You can specify a stock icon or any icon file name.

Usage

In a painter

❖ **To specify an icon for minimization:**

1 Display the Icon tab page of the window's or control's Properties view.

2 Specify an Icon Name by selecting a stock icon from the Stock Icons list box, or use the Browse button to select another icon (*.ICO*) file.

After you have selected an icon, the image is displayed on the Icon tab page.

In scripts

The Icon property takes a string containing the name of the icon file you want to display when the window or control is minimized. You can specify a stock icon or any icon file name.

This example sets the icon for a window to an icon file called *arrow.ico*:

```
w_1.Icon = 'c:\examples\arrow.ico'
```

This example sets the icon for a window to the stock icon Rectangle:

```
w_1.Icon = 'Rectangle!'
```

IgnoreDefaultButton

Applies to

EditMask, MultiLineEdit controls

Description

The IgnoreDefaultButton property specifies whether the Clicked event for the window's default button is triggered when user presses Enter.

When this property is enabled, pressing Enter does not trigger the Clicked event, but instead adds a new line in the control.

When this property is not enabled, pressing Enter does trigger the Clicked event and a new line is *not* added in the control.

Usage

In a painter

❖ **To set the IgnoreDefaultButton property:**

- Select the Ignore Default Button check box on the General page of the control's Properties view.

In scripts

The IgnoreDefaultButton property takes a boolean value. The default is false.

The following example specifies that pressing Enter does not trigger the Clicked event for the window's default button and adds a new line in the MultiLineEdit control instead:

```
mle_1.IgnoreDefaultButton = TRUE
```

IgnorePressure

Applies to

InkEdit, InkPicture controls

Description

A drawing attribute that specifies whether the drawn ink gets wider as the pressure of the pen tip on the tablet surface increases.

Usage

In a painter

❖ **To specify that the pressure of the pen tip should be ignored:**

- Select the IgnorePressure check box on the Ink page in the Properties view.

In scripts

The IgnorePressure property takes a boolean value.

This example sets the IgnorePressure property to true for the control ie_1:

```
ie_1.IgnorePressure = TRUE
```

Increment

Applies to

EditMask controls

Description

When an EditMask control has been defined as a spin control (that is, a control with up and down arrows the user clicks to cycle through predefined values), the Increment property specifies the increment of the spin arrows.

Increment is valid only for numeric and date datatypes. In a date datatype, the increment applies only to the year.

Usage

In a painter

❖ **To set the increment of a spin control:**

1 Select the Spin Control check box on the Mask tab page of the control's Properties view.

The Increment field becomes active.

2 Select the mask datatype from the Type drop-down list.

3 Enter an increment value in the Spin Increment field.

In scripts

The Increment property takes a double. The following line specifies an increment of 10 for an EditMask:

```
em_1.Increment = 10.0
```

Indent

Applies to

TreeView controls

Description

The Indent property specifies how far each level of the TreeView is indented. The numeric value you type is the indentation amount in PowerBuilder units.

Usage

In a painter

❖ **To set the indentation of items in a tree view control:**

- Enter the amount of the indentation, in PowerBuilder units, in the Indentation field of the General page of the control's Properties view.

In scripts

The Indent property takes an integer value. The following script sets an indentation of 100 PowerBuilder units:

```
tv_1.Indent = 100
```

InkAntiAliased

Applies to

InkEdit, InkPicture controls

Description

A drawing attribute that specifies whether the foreground and background colors along the edge of the drawn ink are blended (antialiased) to make the stroke smoother and sharper.

Usage

In a painter

❖ **To specify that the foreground and background colors are not blended:**

- Clear the InkAntiAliased check box on the Ink page in the Properties view.

In scripts

The InkAntiAliased property takes a boolean value.

This example sets the InkAntiAliased property to false for the control ie_1:

```
ie_1.InkAntiAliased = FALSE
```

InkColor

Applies to

InkEdit, InkPicture controls

Description

A drawing attribute that specifies the current ink color. The default color is black.

Usage

In a painter

❖ **To specify a color for the ink drawn in an InkEdit or InkPicture control:**

- Select a color from the InkColor drop-down list on the Ink page in the Properties view.

In scripts

The InkColor property takes a long value.

This example sets the InkColor property to the long value for magenta for the control ip_1:

```
ip_1.InkColor = 16711935
```

This example in the Moved event of a trackbar control sets the InkColor property using the RGB function and the scroll position selected by the user:

```
ip_1.InkColor = RGB(scrollpos/4, scrollpos/16,
scrollpos/64)
```

InkEnabled

Applies to

InkPicture controls

Description

Specifies whether ink collection is enabled.

Usage

In a painter

❖ **To specify that an InkPicture control can collect ink:**

- Select the InkEnabled check box on the Ink page in the Properties view.

In scripts

The InkEnabled property takes a boolean value.

The value of the property is always false on systems that do not have the Tablet PC SDK installed. You must set this property to false before changing the MarginX and MarginY properties, and you should set it to false before closing an application. Before changing this property, make sure the control is not collecting ink by checking the Status property.

This example checks that the ip_1 control is not collecting ink, then disables ink collection:

```
IF ip_1.Status = Idle! THEN
   ip_1.InkEnabled = FALSE
ELSE
   MessageBox("Please try again later",  &
      "Ink is being collected.")
END IF
```

InkHeight

Applies to

InkEdit, InkPicture controls

Description

A drawing attribute that specifies the height of the side of the rectangular pen tip in pixels. The default is 53 pixels. This property has no effect on the ball pen tip.

Usage

In a painter

❖ **To specify the height of the ink drawn in an InkEdit or InkPicture control:**

- Type or select a value in the InkHeight spin control on the Ink page in the Properties view.

In scripts

The InkHeight property takes an integer value.

This example in the Moved event of a trackbar control sets the InkHeight property using the scroll position in the trackbar selected by the user:

```
ip_1.InkHeight = int(scrollpos)
```

InkMode

Applies to

InkEdit controls

Description

Specifies whether ink collection is enabled and whether ink only or ink and gestures are collected.

Usage

In a painter

❖ **To specify the mode of ink collection:**

- Select a value from the InkMode drop-down list on the Ink page in the Properties view.

In scripts

The InkMode property takes a value of the InkMode enumerated variable. Values are:

- CollectInk! – Only ink is collected.
- CollectInkAndGestures! – Ink and gestures are collected (default).
- InkDisabled! – Ink collection is disabled.

The value of the property is always InkDisabled! on systems that do not have an ink recognizer installed.

This example specifies that the InkEdit control will collect ink but ignore gestures:

```
ie_1.InkMode = CollectInk!
```

InkWidth

Applies to

InkEdit, InkPicture controls

Description

A drawing attribute that specifies the width of the pen in pixels. The default is 53 pixels. If the IgnorePressure property is not set, the actual width varies between .5 times the value of the Width property for minimum pressure and 1.5 times its value for maximum pressure.

The pen tip can be a ball or a rectangle. The InkWidth property specifies the diameter of the ball tip and the width of the rectangular tip.

Usage

In a painter

❖ **To specify the width of the ink drawn in an InkEdit or InkPicture control:**

- Type or select a value in the InkWidth spin control on the Ink page in the Properties view.

In scripts

The InkWidth property takes an integer value.

This example sets the InkWidth property for ie_1 to 106:

```
ie_1.InkWidth = 106
```

InputFieldBackColor

Applies to

RichTextEdit controls

Description

The InputFieldBackColor property sets the color for the background of input fields in the RichTextEdit control. This item can also be selected by the user at runtime from the Properties item of the pop-up menu.

Usage

In a painter

❖ **To set the background color of input fields:**

- Select the desired color from the Background Field Color drop-down list on the Document page of the control's Properties view.

  To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

In scripts

The InputFieldBackColor property takes a long (-2 to 16,777,215) that defines the background color for input fields. The value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

This statement makes the input fields red.

```
rte_1.InputFieldBackColor = RGB(255, 0, 0)
```

InputFieldNamesVisible

Applies to

RichTextEdit controls

Description

When the InputFieldNamesVisible property is enabled, the control displays input field names rather than input field values. When this property is not enabled, the RichTextEdit control displays the input field values.

You can set this property in a RichTextEdit control only at runtime using scripts. This property can also be enabled and disabled by the user at runtime from the properties item of the pop-up menu.

Usage

In a painter

The InputFieldNamesVisible property cannot be set at design time for the RichTextEdit control.

In scripts

The InputFieldNamesVisible property takes a boolean value. The following example causes input fields to display data rather than field names:

```
rte_1.InputFieldNamesVisible = FALSE
```

InputFieldsVisible

Applies to

RichTextEdit controls

Description

When the InputFieldVisible property is enabled, input fields appear on the RichTextEdit control

Usage

In a painter

❖ **To make input fields visible:**

- Select the Fields Visible check box on the Document tab page of the control's property page.

In scripts

The InputFieldsVisible property takes a boolean value. The following example specifies that input fields are visible in a RichTextEdit control:

```
rte_1.InputFieldsVisible = TRUE
```

InsertAsText

Applies to

InkEdit controls

Description

Specifies whether the ink is inserted as text or as ink.

Usage

In a painter

❖ **To specify that ink added to the InkEdit control is not converted to text:**

- Clear the InsertAsText check box on the Ink page in the Properties view.

In scripts

The InsertAsText property takes a boolean value. By default, ink is converted to text after a brief pause. If you want the ink to be displayed as ink in the control instead of being converted to text, set the value of the InsertAsText property to false:

```
ie_1.InsertAsText = FALSE
```

Invert

Applies to

Picture and PictureHyperLink controls

Description

If the Invert property is enabled, PowerBuilder displays the picture with its colors inverted. If this property is not enabled, the picture appears in its normal color.

Usage

In a painter

❖ **To invert colors in a picture control:**

- Select the Invert Image check box on the General page of the control's property page.

In scripts

The Invert property takes a boolean value. The following example specifies that a Picture control, p_1, will appear in its normal colors:

```
p_1.Invert = FALSE
```

Italic

Applies to

Controls that display text

Description

Italic is a property of text in a control.

Usage

In a painter

❖ **To italicize all text items in a control:**

- Select the Italic check box on the Font tab page of the control's property page, or select the control and then click the I button on the StyleBar.

❖ **To italicize a text objects in a graph control:**

1. Display the Text tab page of the graph control's Properties view.
2. Select the desired text object from the Text Object list box.
3. Select the Italic check box.

In scripts

The Italic property takes a boolean value. The following example italicizes the text in a StaticText control:

```
st_1.Italic = TRUE
```

This example italicizes the label of the Value axis of a graph control:

```
gr_1.Values.LabelDispAttr.Italic = TRUE
```

Item[ ]

Applies to

ListView, ListBox, PictureListBox, DropDownListBox, DropDownPictureListBox, and Toolbar controls

Description

The Item property array specifies the items in the control. This array is not updated after initialization.

Usage

In a painter

❖ **To add items to a control:**

- Enter the items on the Items tab page of the control's Properties view.

In scripts

The Item[] property is an array of strings, but it is not updated after initialization. Use the AddItem or appropriate InsertItem function instead.

ItemPictureIndex[ ]

Applies to

PictureListBox, DropDownPictureListBox, ListView

Description

The ItemPictureIndex property array identifies the pictures associated with items in the control. This array is not updated after initialization.

Usage

In a painter

❖ **To associate pictures with list items:**

1 Display the Pictures tab page in the control's Properties view and add the pictures to be used in the control to the PictureName list.

   For ListView controls, add pictures to the PictureName lists on the LargePicture, SmallPicture, and State tab pages.

2 Display the Items tab page in the Properties view and add text to the Item list.

3 In the ItemPictureIndex list, add the index number for a picture (from the PictureName lists) on the appropriate lines for the items with which you want to associate pictures.

In scripts

You add pictures to controls with the AddPicture function and add items to these controls with the AddItem or InsertItem functions. You use picture indexes in the AddItem and InsertItem functions to associate pictures with the items. See "Using Lists in a Window" in *Application Techniques* for more information.

Label

Applies to

grAxis objects in Graph controls, ListViewItem objects, TreeViewItem objects

Description

Within graphs The Label property specifies the label of an axis of the graph.

Within ListView and TreeView controls The Label property specifies the label associated with a ListViewItem or TreeViewItem object.You cannot manipulate items in TreeView controls in a painter. You must write scripts to add items to a TreeView.

Usage

In a painter

❖ **To specify an Axis label in a graph control:**

1 Display the Axis tab page of the graph control's Properties view.

2 Select the desired axis from the Axis drop-down list.

3 Enter the label text in the Label text field.

❖ **To specify labels for items in a ListView control:**

1 Display the Items tab page of the ListView control's Properties view.

2 For each item, enter label text in the appropriate Text field.

In scripts

The Label property takes a string value. The following example sets text for the label on the Values axis of graph gr_1.

```
gr_1.Values.Label = 'Lawsuits per 1000'
```

To add or insert an item with a label into a ListView control, use the AddItem or InsertItem functions. For example, this line adds an item to ListView control lv_1, specifying the label and picture index for the item:

```
lv_1.AddItem ( "Oranges", 1)
```

To change the label, get the item from the ListView and set the item's Label property:

```
ListViewItem lvi
lv_1.GetItem(4, lvi)
lvi.Label = "Apples"
lv_1.SetItem(4, lvi)
```

To add or insert items in a TreeView control, use the InsertItem, InsertItemFirst, InsertItemLast, or InsertItemSort functions.

For more information, see "Using TreeView Controls" and "Using ListView Controls" in *Application Techniques*.

LabelWrap

Applies to

ListView controls

Description

When the LabelWrap property is enabled, long ListView item labels wrap in a large icon view. If LabelWrap is not enabled, labels are displayed on a single line. LabelWrap does not apply to report, list, or small icon views.

Usage

In a painter

❖ **To enable label wrap for a ListView control:**

- Select the Label Wrap check box on the General page of the control's Properties view.

In scripts

The LabelWrap property takes a boolean value. The following line enables word wrapping of labels in a ListView:

```
lv_1.LabelWrap = TRUE
```

LargePictureHeight

Applies to

ListView controls

Description

The LargePictureHeight property specifies the display height of all the pictures in the Large Icon view of the ListView control. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the height of the first picture in the array as the height for all the pictures. The other choices in the painter, 16 and 32, are standard pixel heights for icons.

Usage

In a painter

❖ **To set the large picture height:**

- Select a value from the Height drop-down list on the Large Picture tab page of the control's Properties view.

In scripts

The LargePictureHeight property takes an integer value. This value can be set only before the first call to the AddLargePicture function or after calling DeleteLargePictures. If this value is set to 0, then the size of the first picture is used to set the size of large pictures.

The following line sets the height for large pictures in a ListView to 32 pixels:

```
lv_1.LargePictureHeight = 32
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

LargePictureMaskColor

Applies to

ListView controls

Description

The mask color is the color in the picture that is transparent when the picture is displayed.

Select the color to mask newly added user-defined bitmaps. In scripts, you can change the mask color before adding each picture. Each image uses the mask color that was in effect when it was added.

Usage

In a painter

❖ **To specify a picture mask color:**

- Select a color from the Picture Mask Color drop-down list on the Large Picture tab page of the control's Properties view.

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

In scripts

The LargePictureMaskColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color. This property is used when each bitmap is added and, therefore, can be changed between AddLargePicture calls.

The LargePictureMaskColor value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for a particular color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the mask color for user-defined bitmaps in a ListView:

```
lv_1.LargePictureMaskColor = RGB(255, 255, 0)
```

LargePictureName[ ]

Applies to

ListView controls

Description

PowerBuilder stores ListView images in several indexed arrays of images. You can associate an image with a specific ListView item when you create a ListView in the painter or use the AddItem and InsertItem functions at execution time.

You identify a specific image by its index number. Because the same index number refers to both the large picture and the small picture for the item (depending on which view is selected), you need to make sure the images for each position in the array are compatible.The type of image used is determined by the value of the View property of the control.

Usage

In a painter

❖ **To specify images for the Large Icon view**

1 Select the Large Picture tab page from the ListView control's Properties view.

2 Do one of the following:

- In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
- Use the Browse button.
- Select one or more pictures from the Stock Pictures list.

The order of the picture names specified here should match the picture name order used for the Small Icon view.

3 Use the row numbers from this Picture Name list to specify the Picture Index for each List View Item on the Items tab page.

In scripts

The LargePictureName property takes a string value. You cannot use the LargePictureName property to update the image list during execution. Use the AddLargePicture function to add large pictures to a ListView control. For example:

```
lv_1.AddLargePicture("c:\ArtGal\bmps\celtic.bmp")
```

When you add a large picture to a ListView control, it is given the next available picture index in the ListView.

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

LargePictureWidth

Applies to

ListView controls

Description

The LargePictureWidth property specifies the display width of all the pictures in the Large Icon view of the ListView control. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the width of the first picture in the array as the width for all the pictures. The other choices in the painter, 16 and 32, are standard pixel widths for icons.

Usage

In a painter

❖ **To set the large picture width:**

- Select a value from the Width drop-down list on the Large Picture tab page of the control's Properties view.

In scripts

The LargePictureWidth property takes an integer value. This value can be set only before the first call to the AddLargePicture function or after calling DeleteLargePictures. If this value is set to 0, then the size of the first picture is used to set the size of large pictures.

The following line sets the width for large pictures in a ListView to 32 pixels:

```
lv_1.LargePictureWidth = 32
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

LayoutRTL

Applies to

ListView and TreeView controls

Description

The LayoutRTL property specifies that the layout of the control should be a mirror image of the standard layout. Scroll bars display at the left of the control. In a ListView, icons are right justified. In a TreeView, the root-level icon is right justified and its label displays to the left of the icon. Lower levels are indented from the right. Unlike the RightToLeft property, which affects the display of characters, the LayoutRTL property does not require an operating system that supports right-to-left display. Values are:

- TRUE – Elements in the control are right justified.

- FALSE – Elements in the control are left justified (default).

For best results, set this property in the painter so that you can see its effect. Setting this property at runtime can have unexpected results.

Usage

In a painter

❖ **To set the LayoutRTL property:**

- Select the RTL Layout check box on the Other page in the control's Properties view.

In scripts

The LayoutRTL property takes a boolean value.

The following line sets the LayoutRTL property of a ListView control to true:

```
lv_1.LayoutRTL = TRUE
```

The following lines determine the value of the LayoutRTL property for a TreeView control:

```
boolean bRTL
bRTL = tv_1.LayoutRTL
```

LeftMargin

Applies to

RichTextEdit controls

Description

The LeftMargin property specifies the size in inches of the left margin on the printed page.

Usage

In a painter

❖ **To set the left margin:**

- Enter the desired size in inches in the Left Margin field of the Document tab page of the RichTextEdit control's property page.

In scripts

The LeftMargin property takes a long value. The following line sets the left margin of a RichTextEdit to 1 inch:

```
rte_1.LeftMargin = 1
```

LeftText

Applies to

CheckBox, RadioButton controls

Description

If the LeftText property is enabled, the text for a CheckBox or RadioButton appears to the left of the button. If LeftText is not enabled, the text appears to the right of the control. You can also specify left or right alignment with the left and right StyleBar buttons.

Usage note

When the LeftText property is enabled and you align multiple CheckBoxes or RadioButtons to the left, PowerBuilder may align the text but not the boxes or buttons. This is because PowerBuilder aligns the complete control.

Usage

In a painter

❖ **To place text to the left:**

- Select the Left Text check box on the General page of the control's Properties view.

In scripts

The LeftText property takes a boolean value. The following line puts the text for a CheckBox to the left of the box:

```
cbx_1.LeftText = TRUE
```

Legend

Applies to

Graph controls

Description

The value of the Legend property specifies the placement of the graph's legend, or that there should be no legend.

Usage

In a painter

❖ **To set the location of the legend:**

- Select the desired location from the Legend drop-down list on the General page of the control's Properties view.

In scripts

The Legend property takes a value of the grLegendType enumerated datatype. The following example sets the location of the legend to below the graph:

```
gr_1.Legend = AtBottom!
```

Limit

Applies to

DropDownListBox, DropDownPictureListBox, EditMask, InkEdit, MultiLineEdit, SingleLineEdit controls

Description

The Limit property controls the number of characters the user can enter in the control.

Usage

In a painter

❖ **To set the maximum number of characters allowed:**

- Type the number of characters that the user can enter in the control in the Limit field of the General page of the control's Properties view.

  0 indicates an unlimited number of characters.

In scripts

The Limit property takes an integer value.

The following example sets 256 as the maximum number of characters for the MultiLineEdit mle_1:

```
mle_1.Limit = 256
```

LineColor

Applies to

Line, Oval, Rectangle, RoundRectangle controls

Description

The line color is the color for the border and the lines in the control's fill pattern.

Usage

In a painter

❖ **To change the line color:**

- Select a color from the Line Color drop-down list on the General page of the control's property sheet, or select a color from the Background Color option on the Style Bar.

Using the StyleBar lets you change the line color for several selected objects at the same time. To add your own colors to the color drop-down list, select Design>Custom Colors before opening the Properties view.

In scripts

The LineColor property takes a long value. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following statement specifies red as the line color for a Rectangle:

```
r_1.LineColor = RGB(255,0,0)
```

LinesAtRoot

Applies to

TreeView controls

Description

When LinesAtRoot is enabled, PowerBuilder connects all root items in a TreeView with lines.

Usage

In a painter

❖ **To specify that root items in a TreeView are connected:**

- Select the Lines At Root check box on the General page of the control's property page.

In scripts

The LinesAtRoot property takes a boolean value.

The following line specifies that all root items in a TreeView are connected:

```
tv_1.LinesAtRoot = TRUE
```

LinesPerPage

Applies to

User objects and windows

Description

The LinesPerPage property determines the number of lines on a page for scrolling purposes. PowerBuilder multiplies Units Per Line by Lines Per Page to determine the number of PowerBuilder units to scroll the window vertically when the user clicks in the scroll bar.

For information on calculating LinesPerPage and UnitsPerLine, see Scrolling in windows and user objects on page 591.

To control the horizontal scroll bar in a window or user object, use the UnitsPerColumn and ColumnsPerPage properties.

Usage

In a painter

❖ **To set the LinesPerPage property:**

- Enter the desired number (between 1 and 100) in the Lines Per Page option on the Scroll tab page of the window's Properties view.

In scripts

The LinesPerPage property takes an integer value between 1 and 100. The following line sets LinesPerPage for a window to 20:

```
This.LinesPerPage = 20
```

LineStyle

Applies to

Line, Oval, Rectangle, RoundRectangle controls

Description

The LineStyle property specifies the appearance of a line object or the border around other drawing objects.

Usage

In a painter

❖ **To set the line style:**

- Select a line style from the Line Style drop-down list on the General page of the control's Properties view.

In scripts

The LineStyle property takes a value of the LineStyle enumerated datatype. The following example sets a dashed line for a Rectangle:

```
r_1.LineStyle = Dash!
```

LinkUpdateOptions

Applies to

OLE controls

Description

The LinkUpdateOptions property specifies how a linked object in an OLE control is updated. There are two options: automatic and manual. If automatic is chosen, the link is updated when the object is opened and whenever the object changes in the server application. If manual is chosen, the link is not updated.

Usage

In a painter

❖ **To set the Link Update option:**

- Choose Automatic or Manual from the Link Update drop-down list on the General page of the control's Properties view.

In scripts

The LinkUpdateOptions property takes a value of the omLinkUpdateOptions enumerated datatype.

The following example sets the OLE control's update option to automatic:

```
ole_1.LinkUpdateOptions = LinkUpdateAutomatic!
```

LiveScroll

Applies to

DataWindow controls

Description

When the LiveScroll property is enabled, the rows in the DataWindow control scroll while the user is moving the scroll box. If this property is not enabled, the rows do not scroll until the user releases the scroll box.

Usage

In a painter

❖ **To enable live scrolling in a DataWindow control:**

- Select the Live Scrolling check box on the General page of the control's Properties view.

In scripts

The LiveScroll property takes a boolean value.

The following example allows scrolling while the user moves the scroll box in the DataWindow control:

```
This.LiveScroll = TRUE
```

LogFileName

Applies to

MLSynchronization and MLSync objects

Description

Specifies the full name of the log file for the synchronization process.

Usage

At design time, you can enter a value for the log file name on the Logging tab of the Properties view for an MLSync object. At runtime, application users can enter a value for the log file name on the Settings tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can change the log file name in script as follows:

```
mySync_1.logfilename = "c:\documents\mylogfile.txt"
```

LogOpts

Applies to

MLSynchronization and MLSync objects

Description

Specifies dbmlsync options to control logging output.

Usage

At design time, you can enter a value for LogOpts in the Log Options text box on the Logging tab of the Properties view for an MLSync object. At runtime, application users can enter a value for LogOpts in the Logging text box on the Settings tab page of the default synchronization options window generated by the MobiLink wizard.

Logging options are indicated with a short text description on the MobiLink Client Logging Options page of the MobiLink wizard.

In scripts

You can change the logging options in script as follows:

```
mySync_1.LogOpts = "-v+"
```

The `-v+` value logs all synchronization information except the connection string and the MobiLink password.

MajorGridLine

Applies to

grAxis objects in Graph controls

Description

The major grid for an axis extends from the tick marks on the axis across the data area of the graph. The minor grid lines fall between the tick marks and display when the number of divisions is 2 or greater.

The MajorGridLine property specifies the line style for the major grid.

Usage

In a painter

❖ **To set the line style for a major grid line:**

1 Display the Axis tab page of the graph control's Properties view and select the desired axis from the Axis drop-down list.

2 Select the desired line style from the MajorGridLine drop-down list in the Major Divisions group.

In scripts

The MajorGridLine property takes a value of the LineStyle enumerated datatype.

This example specifies a dashed line for the major grid line on the Value axis of Graph gr_1:

```
gr_1.Value.MajorGridLine = Dash!
```

MajorDivisions

Applies to

grAxis objects in Graph controls

Description

The MajorDivisions property specifies the number of divisions or ticks on the axis, not counting the origin point.

The default value of 0 means the graph uses a MajorDivision value optimized for the data and suppresses MinorDivision ticks.

Usage

In a painter

❖ **To specify the number of major divisions on an axis:**

1 Display the Axis tab page from the graph's Properties view and select the desired axis from the Axis drop-down list.

2 Use the spin control in the MajorDivisions field of the Major Divisions group to specify the desired number of divisions.

In scripts

The MajorDivisions property takes an integer specifying the number of major divisions on an axis.

The following example sets 10 ticks on the major grid of the Values axis of a graph.

```
gr_1.Values.MajorDivisions = 10
```

MajorTic

Applies to

grAxis objects in Graph controls

Description

The MajorTic property specifies how ticks overlap the axis for the major grid. Ticks can be placed on the inside of the axis line, on the outside, or straddling it; or there can be no ticks visible.

Usage

In a painter

❖ **To specify the type of major tick marks:**

1 Display the Axis tab page of the control's Properties view and select the desired axis from the Axis drop-down list.

2 Select the desired type of tick mark from the MajorTick drop-down list box in the Major Division group.

In scripts

The MajorTic property takes a value of the grTicType enumerated datatype.

The following line sets ticks on the major grid to straddle the grid:

```
gr_1.Values.MajorTic = Straddle!
```

Map3DColors

Applies to

Picture, PictureHyperlink, and PictureButton controls

Description

Maps the silver and other gray colors in the bitmap associated with the control to the button highlight, button face, or button shadow colors set in the Windows control panel. When this property is false (the default), the control uses the standard PowerBuilder button colors defined in the bitmap.

Use this feature if you want to place a control containing a picture on a window and have the picture blend in with the background color of the window when the window's background is using Button Face for a 3D effect. The control's picture then takes on the 3D colors the user has selected on the Appearance page of the Display Properties dialog box in the Windows control panel.

The window's background must be set to Button Face. To make the image blend in with the window, give it a background color in the range between RGB(160,160,160) and RGB(223,223,223), such as silver. Lighter shades of gray map to the button highlight color and darker shades to the button shadow color.

This option can affect other colors used in the bitmap. It does not affect the control's border settings, and it has no effect if there is no image associated with the control.

Usage

In a painter

❖ **To set 3DColor mapping:**

- Select the 3D Color check box on the General page of the control's Properties view.

In scripts

The Map3DColors property takes a boolean value. The following example sets 3D color mapping for a PictureButton:

```
pb_1.Map3DColors = TRUE
```

Mask

Applies to

EditMask controls

Description

The Mask property controls the characters the user can enter in the control and also the formatting of the characters. You must use special characters to define the mask, depending on the mask type defined with the MaskDataType property.

PowerBuilder supports six mask types:

- DateMask!
- DateTimeMask!
- DecimalMask!

- NumericMask!
- StringMask!
- TimeMask!

Characters that have special significance for each mask type display in the Mask drop-down list. Characters that do not have special meaning for the format appear as is in the EditMask control.

For most mask types, the special characters you can use in a mask are the same as those you can use in a display format. For more information about using each kind of display format, see Chapter 4, About Display Formats and Scrolling. For more information about defining display formats, see the *User's Guide.*

The special characters you can use in string edit masks are different from those you can use in string display formats.

Table 3-2: Special characters for string edit masks

Character	Meaning
!	Uppercase – displays all characters with letters in uppercase
^	Lowercase – displays all characters with letters in lowercase
#	Number – displays only numbers
a	Alphanumeric – displays only letters and numbers
X	Any character – displays all characters

If you use the "#" or "a" special characters in a mask, then Unicode characters, spaces, and other characters that are not alphanumeric do not display.

Usage

In a painter

❖ **To specify an edit mask:**

1 Display the Mask tab page of the control's Properties view.

2 Select the mask datatype from the MaskDataType drop-down list.

3 Type the mask characters in the Mask field, or click the right arrow at the end of the Mask field and select one or more of the mask character examples displayed in the pop-up menu.

The pop-up menu examples change based on the mask datatype you selected in the MaskDataType list.

In scripts

The Mask property takes a string value and can be used to obtain the value of a mask. It cannot be used to set the value.

The following example uses the SetMask function to set the datatype and string format for a mask, and then uses the Mask property to obtain the value of the string format. The mask specifies that the first letter in the string is displayed in uppercase and the next nine characters in lowercase. If the string has more than ten characters, they do not display:

```
string ls_mask

em_1.SetMask(StringMask!, '!^^^^^^^^^')
ls_mask = em_1.Mask
```

MaskDataType

Applies to

EditMask controls

Description

This property specifies the datatype of the control. The special characters used to define the mask differ depending on the datatype of the mask control.

Usage

In a painter

❖ **To select the mask datatype:**

- Select the desired type from the Type drop-down list box in the Options group on the Mask tab.

  The examples of special characters displayed in the Masks field change to show the characters relevant to the selected mask datatype.

In scripts

The MaskDataType property takes a value of the MaskDataType enumerated datatype and can be used to obtain the datatype of a mask. It cannot be used to set the datatype.

The following example uses the SetMask function to set the datatype and date format for a mask, and then uses the Mask property to obtain the value of the date format:

```
MaskDataType l_mdt

em_1.SetMask(DateMask!, 'mm/dd/yy')
l_mdt = em_1.MaskDataType
```

MaxBox

Applies to

DataWindow controls, Windows

Description

The MaxBox property specifies whether a Maximize box is displayed on the control's title bar.

Usage

In a painter

❖ **To display a Maximize box on a DataWindow control:**

1 Select the TitleBar check box on the General page of the control's Properties view.

2 Select the MaxBox check box on the General page.

❖ **To display a Maximize box in a window:**

- If the MaxBox check box is grayed out, select the Title Bar check box on the General page of the window's Properties view, then select the MaxBox check box.

  For certain types of windows, having a title bar is not the default and therefore the MaxBox check box can be grayed out until you select the TitleBar check box.

In scripts

The MaxBox property takes a boolean value.

The following example specifies that the DataWindow control should have a title bar with a Maximize box:

```
dw_1.TitleBar = TRUE
dw_1.MaxBox = TRUE
```

MaximumValue

Applies to

grAxis objects in Graph controls

Description

The MaximumValue property specifies the maximum value for an axis when the axis datatype is numeric. This property is not used if the Autoscale property is enabled.

Usage

In a painter

❖ **To set the maximum value of an axis with a numeric datatype:**

1 Display the Axis tab page of the graph's Properties view and select the desired axis from the Axis drop-down list.

2 Make sure that the Autoscale check box is not checked.

3 Select adtDouble! from the DataType drop-down list.

4 Specify the desired maximum numeric value in the MaximumValue field.

This value should be larger than the maximum data value being graphed.

In scripts

The MaximumValue property takes a double value.

The following line sets a maximum value for an Axis with a datatype of double.

```
gr_1.Values.DataType = AdtDouble!
gr_1.Values.MaximumValue = 500000.00
```

MaxDate

Applies to

DatePicker controls

Description

Specifies the maximum date that the user can select in the calendar.

Usage

In a painter

❖ **To set the MaxDate property:**

- Select a date from the drop-down calendar or type a date in the MaxDate field on the General page in the Properties view.

In scripts

The MaxDate property takes a Date value. The default is December 31, 2999. You can change this setting to restrict the range of dates a user can select. This example sets MaxDate to June 30, 2007:

```
dp_1.MaxDate = Date("2007/06/30")
```

MaxPosition

Applies to

HProgressBar, VProgressBar, HScrollBar, VScrollBar, HTrackBar, VTrackBar controls

Description

The MaxPosition property specifies the value of the Position property when the progress indicator, scroll box, or slider is at the bottom of the vertical control or the right edge of the horizontal control. For a progress bar, this value can be different from the end of the control's range, set with the SetRange function.

Usage

In a painter

❖ **To specify the maximum position of the progress indicator, scroll box, or slider:**

- Enter an integer value into the Max Position field of the General tab of the control's Properties view.

In scripts

The MaxPosition property takes an integer value.

The following example specifies that the value of the Position property is 120 when a scroll box is in the maximum position:

```
vsb_1.MaxPosition = 120
```

MaxSelectCount

Applies to

MonthCalendar controls

Description

Specifies the maximum number of days the user can select in the calendar.

Usage

In a painter

❖ **To set the MaxSelectCount property:**

- Select a number from the MaxSelectCount spin control on the General page in the Properties view.

In scripts

The MaxSelectCount property takes an integer value between 1 and 360. The default is 31. You can change this setting to enable users to select fewer or more than 31 days in the calendar and before calling the SetSelectedRange function if you want to set a different limit. This example sets MaxSelectCount to 7:

```
mc_1.MaxSelectCount = 7
```

MaxValDateTime

Applies to

grAxis objects in Graph controls

Description

The MaxValDateTime property specifies the maximum value for an axis when the axis datatype is date or time. This property is not used if the Autoscale property is enabled.

Usage

In a painter

❖ **To set the maximum value of an axis with a date or time datatype:**

1 Display the Axis tab page of the graph's Properties view and select the desired axis from the Axis drop-down list.

2 Make sure that the Autoscale check box is not checked.

3 Select adtDate!, adtTime!, or adtDateTime! from the DataType drop-down list.

4 Specify the desired maximum date or time value in the MaximumValue field.

This value should be larger than the maximum data value being graphed.

In scripts

The MaxValDateTime property takes a value of the DateTime datatype.

The following example sets the MaxValDateTime property for an Axis with a datatype of date:

```
gr_1.Values.DataType = AdtDate!
gr_1.Values.MaxValDateTime = 12/31/1999
```

MenuName

Applies to

Windows

Description

The MenuName property specifies the menu object that is the menu for the window.

Usage

In a painter

❖ **To specify a menu:**

- Enter a menu name in the Menu Name field on the General page of the window's Properties view, or use the Browse button to choose a menu object from the current or another PBL.

In scripts

The MenuName property takes a string containing the name of a menu object. PowerBuilder uses it internally to identify the menu. Do not change this property in a script. Instead, use the ChangeMenu or PopMenu functions to display a menu.

MinBox

Applies to

DataWindow controls, Windows

Description

The MinBox property specifies whether a Minimize box is displayed on the control's title bar.

Usage

In a painter

❖ **To display a Minimize box on a DataWindow control:**

1 Select the TitleBar check box on the General page of the control's Properties view.

2 Select the MinBox check box on the General page.

❖ **To display a Minimize box in a window:**

- If the MinBox check box is grayed out, select the TitleBar check box on the General page of the window's Properties view, then select the MinBox check box.

  For certain types of windows, having a title bar is not the default and therefore the MinBox check box can be grayed out until you select the TitleBar check box.

In scripts

The MinBox property takes a boolean value.

The following example specifies that the DataWindow control should have a title bar with a Minimize box:

```
dw_1.TitleBar = TRUE
dw_1.MinBox = TRUE
```

MinDate

Applies to

DatePicker controls

Description

Specifies the minimum date that the user can select in the calendar.

Usage

In a painter

❖ **To set the MinDate property:**

- Select a date from the drop-down calendar or type a date in the MinDate field on the General page in the Properties view.

In scripts

The MinDate property takes a Date value. The default is January 1, 1800. You can change this setting to restrict the range of dates a user can select. This example sets MinDate to September 1, 2006:

```
dp_1.MinDate = Date("2006/09/01")
```

MinimumValue

Applies to

grAxis objects in Graph controls

Description

The MinimumValue property specifies the minimum value for an axis when the axis datatype is numeric. This property is not used if the Autoscale property is enabled.

Usage

In a painter

❖ **To set the minimum value of an axis with a numeric datatype:**

1. Display the Axis tab page of the graph's Properties view and select the desired axis from the Axis drop-down list.
2. Make sure that the Autoscale check box is not checked.
3. Select adtDouble! from the DataType drop-down list.
4. Specify the desired minimum numeric value in the MinimumValue field.

   This value should be smaller than the minimum data value being graphed.

In scripts

The MinimumValue property takes a double value.

The following line sets a minimum value for an Axis with a datatype of double:

```
gr_1.Values.DataType = AdtDouble!
gr_1.Values.MinimumValue = 0.00
```

MinMax

Applies to

EditMask controls

Description

The value of the MinMax property specifies the minimum and maximum values allowed when the EditMask functions as a spin control.

You can specify minimum and maximum values only for date and numeric datatypes. For dates, enter a full date (for example, 1/1/2003), although the minimum and maximum values affect only the year. The user can scroll freely through the days and months.

Usage

In a painter

❖ **To set the minimum and maximum spin values:**

1 Select the Spin Control check box on the Mask tab page of the control's Properties view.

2 Enter minimum and maximum values in the Min and Max fields in the Spin Range group.

In scripts

The MinMax property takes a string value. The values are separated with two tildes (~~).

The following example sets the minimum and maximum spin for an Edit Mask em_1:

```
em_1.MinMax = ("100 ~~ 10000")
```

MinorDivisions

Applies to

grAxis objects in Graph controls

Description

The MinorDivisions property specifies the number of spaces between major ticks. To see minor ticks, specify a value of two or greater because the last minor tick is overlaid by the next major tick.

The default value of 0 in the MajorDivisions field means the graph uses a MajorDivision value optimized for the data and suppresses MinorDivision ticks.

Usage

In a painter

❖ **To specify the number of minor divisions on an axis:**

1 Display the Axis tab page from the graph's Properties view and select the desired axis from the Axis drop-down list.

2 Use the spin control in the MinorDivisions field of the Minor Divisions group to specify the desired number of divisions.

In scripts

The MinorDivisions property takes an integer specifying the number of minor divisions on an axis.

The following example sets 10 ticks on the minor grid of the Values axis of a graph.

```
gr_1.Values.MinorDivisions = 10
```

MinorGridLine

Applies to

grAxis objects in Graph controls

Description

The minor grid for an axis extends from the tick marks on the axis across the data area of the graph. The minor grid lines fall between the tick marks and display when the number of divisions is two or greater.

The MinorGridLine property specifies the line style for the minor grid.

Usage

In a painter

❖ **To set the line style for a minor grid line:**

1 Display the Axis tab page of the graph control's Properties view and select the desired axis from the Axis drop-down list.

2 Select the desired line style from the MinorGridLine drop-down list in the Minor Divisions group.

In scripts

The MinorGridLine property takes a value of the LineStyle enumerated datatype.

This example specifies a dotted line for the minor grid line on the Value axis of Graph gr_1:

```
gr_1.Value.MinorGridLine = Dot!
```

MinorTic

Applies to

grAxis objects in Graph controls

Description

The MinorTic property specifies how ticks overlap the axis for the minor grid. Ticks can be placed on the inside of the axis line, on the outside, or straddling it; or there can be no ticks visible.

Usage

In a painter

❖ **To specify the type of minor tick marks:**

1 Display the Axis tab page of the control's Properties view and select the desired axis from the Axis drop-down list.

2 Select the desired type of tick mark from the MinorTicks drop-down list box in the Minor Division group.

In scripts

The MinorTic property takes a value of the grTicType enumerated datatype.

The following line sets ticks on the minor grid to outside the grid:

```
gr_1.Values.MinorTic = Outside!
```

MinPosition

Applies to

HProgressBar, VProgressBar, HScrollBar, VScrollBar, HTrackBar, VTrackBar controls

Description

The MinPosition property specifies the value of the Position property when the progress indicator, scroll box, or slider is at the top of the vertical control or the left edge of the horizontal control. For a progress bar, this value can be different from the start of the control's range, set with the SetRange function.

Usage

In a painter

❖ **To specify the minimum position of the progress indicator, scroll box, or slider:**

- Enter an integer value into the Min Position field of the General tab of the control's Properties view.

In scripts

The MinPosition property takes an integer value.

The following example specifies that the value of the Position property will be 0 when a scroll box is in the minimum position:

```
vsb_1.MinPosition = 0
```

MinValDateTime

Applies to

grAxis objects in Graph controls

Description

The MinValDateTime property specifies the minimum value for an axis when the axis datatype is date or time. This property is not used if the Autoscale property is enabled.

Usage

In a painter

❖ **To set the minimum value of an axis with a date or time datatype:**

1. Display the Axis tab page of the graph's Properties view and select the desired axis from the Axis list.
2. Make sure that the Autoscale check box is not checked.
3. Select adtDate!, adtTime!, or adtDateTime! from the DataType drop-down list.
4. Specify the desired minimum date or time value in the MinimumValue field.

   This value should be smaller than the minimum data value being graphed.

In scripts

The MinValDateTime property takes a value of the DateTime datatype.

The following example sets the MinValDateTime property for an Axis with a datatype of date:

```
gr_1.Values.DataType = AdtDate!
```

```
gr_1.Values.MinValDateTime = 01/31/1900
```

MLPass

Applies to

MLSynchronization, MLSync, and SyncParm objects

Description

Specifies the MobiLink password passed to the synchronization server.

Usage

At design time, you can enter a value for MLPass on the General tab of the Properties view for an MLSync object. At runtime, application users can enter a value for MLPass in the MLPassword text box on the Subscriptions tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can set the MobiLink password in script as follows:

```
mySync_1.MLPass = "myMLPassword"
```

You can also set a new MobiLink password with the SetNewMobiLinkPassword function.

MLServerVersion

Applies to

MLSynchronization and MLSync objects

Description

Specifies the MobiLink server version. You can use 9, 10, or 11 as the server version.

Usage

This is a required property of the synchronization object. If the property value has not been set before a synchronize call, the synchronization fails and an error string is saved to the synchronization object's ErrorText property.

At design time, you can enter a value for MLServerVersion on the General tab of the Properties view for an MLSync object.

In scripts

You can set the MobiLink server version in script as follows:

```
mySync_1.MLServerVersion = 11
```

MLUser

Applies to

MLSynchronization, MLSync, and SyncParm objects

Description

Specifies the user name passed to the MobiLink synchronization server.

Usage

This is a required property of the synchronization object. If the property value has not been set before a synchronize call, the synchronization fails and an error string is saved to the synchronization object's ErrorText property.

At design time, you can enter a value for MLUser on the General tab of the Properties view for an MLSync object. At runtime, application users can enter a value for MLUser in the MLUser text box on the Subscriptions tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can set the MobiLink password in script as follows:

```
mySync_1.MLUser = "me"
```

Modified

Applies to

InkEdit, RichText controls

Description

Specifies whether the text in the control has been modified since it was opened or last saved. Modified is the control's "dirty" flag, indicating that the control is in an unsaved state.

Usage

The value of the Modified property controls the Modified event. If the property is false, the event occurs when the first change occurs to the contents of the control. The change also causes the property to be set to true, which suppresses the Modified event. You can restart checking for changes by setting the property back to false.

In scripts

The Modified property takes a boolean value. The following example sets the Modified property of the InkEdit control ie_1 to false so that the Modified event is enabled:

```
ie_1.Modified = FALSE
```

MonthBackColor

Applies to

MonthCalendar controls

Description

The MonthBackColor property defines the color to be used for the background of a month in the calendar.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select a color from the MonthBackColor drop-down list on the General page in the Properties view.

In scripts

The MonthBackColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color of the month or months in a calendar. The MonthBackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the background color for months:

```
mc_1.MonthBackColor = RGB(255, 255, 0)
```

MultiSelect

Applies to

ListBox, PictureListBox controls

Description

The MultiSelect property specifies whether users can select multiple items in the list box at one time. When it is enabled, users can select multiple items by clicking them. When MultiSelect is not enabled, users cannot select multiple items at once.

If MultiSelect and ExtendedSelect are both enabled, then the behavior of ExtendedSelect takes precedence. For ExtendedSelect, the user must press Shift or Ctrl when clicking additional items.

Usage

In a painter

❖ **To enable multiple selections from the list:**

- Select the MultiSelect check box on the General page of the control's Properties view.

In scripts

The MultiSelect property takes a boolean value. The following example allows multiple selections in the ListBox lb_1:

```
lb_1.MultiSelect = TRUE
```

Multiline

Applies to

Tab controls

Description

When Multiline is enabled, the tabs can appear in more than one row if there is not room for all the tabs in a single row. When Multiline is not enabled, a dual arrow control appears to allow the user to scroll to tabs that do not fit.

Usage

In a painter

❖ **To enable multiline display of tabs:**

- Select the Multiline check box on the General page of the control's Properties view.

In scripts

The Multiline property takes a boolean value. The following line allows tabs to be arranged in multiple rows when necessary:

```
tab_1.Multiline = TRUE
```

ObjectRevision

Applies to

MLSynchronization and MLSync objects

Description

Specifies the build number for synchronization property values that are stored in the client registry.

Usage

When you deploy a new version of an application that includes a wizard-generated synchronization object, you can make sure that values for synchronization object properties are written to the application user's registry by incrementing the value of the ObjectRevision property. Code in the synchonization object's Contructor event checks the value of this property against the value stored during a previous synchronization.

If the value of the ObjectRevision property is higher than the value stored in the registry, property values of the synchronization object are written to the registry, replacing any values previously stored there. The property values written to the registry are: AdditionalOpts, DownloadOnly, ExtendedOpts, Host, LogFileName, LogOpts, MLServerVersion, MLUser, ObjectRevision, Port, Publication, UploadOnly, UseLogFile, and UseWindow.

Secured properties such as AuthenticateParms, DBPass, and EncryptionKey are never written to the registry.

In a painter

On the Settings tab of the Properties view for an MLSync object, type the value you want for the ObjectRevision property in the Object Revision Number text box.

In scripts

The following code is similar to code in the Constructor event of the MLSync object generated by the MobiLink synchronization wizard. It checks the ObjectRevision value against the revision number stored in the registry. If the registry value is less than the value of the ObjectRevision value, the object's synchronization properties are written to the registry. Otherwise, the synchronization object retrieves the values stored in the registry:

```
long   rc = 1
long   RegistryRevision
if this.ObjectRevision > 0 and &
  this.SyncRegistryKey <> "" then
    RegistryRevision = &
       this.GetObjectRevisionFromRegistry()
    if RegistryRevision < this.ObjectRevision then
       rc = this.SetSyncRegistryProperties()
    else
       rc = this.GetSyncRegistryProperties()
    end if
end if
return rc
```

OpenAnimation

Applies to

Window controls

Description

Specifies an optional animation effect that displays when the window opens.

Usage

The OpenAnimation property takes a value of the WindowAnimationStyle enumerated variable. For "slide" values, the whole window appears to slide into view from the direction selected. For "roll" values, the window is painted from the direction selected. Values are:

- NoAnimation! (default) – The window displays with no animation.
- TopSlide! – The window slides from the top to the bottom of its extent.
- BottomSlide! – The window slides from the bottom to the top of its extent.
- LeftSlide! – The window slides from the left to the right of its extent.
- RightSlide! – The window slides from the right to the left of its extent.
- TopRoll! – The window rolls from the top to the bottom of its extent.
- BottomRoll! – The window rolls from the bottom to the top of its extent.
- LeftRoll! – The window rolls from the left to the right of its extent.
- RightRoll! – The window rolls from the right to the left of its extent.
- FadeAnimation! – The window fades in.
- CenterAnimation! – The window expands from the center.

You can modify the animation properties at any time and use them for any window type. They are most often used in pop-up windows. FadeAnimation! can be used only in top-level windows. It does not work in child windows. In MDI applications, you cannot use FadeAnimation! for sheet windows. Fading affects the transparency of the window, and sheet windows in MDI applications always inherit the transparency of the frame window.

Some controls, such as InkEdit, InkPicture, and RichTextEdit controls, may not paint properly when you use animation features. For example, if you place an InkPicture control on a window, the image in the control does not display when the animation completes until the control is clicked.

While the animation executes, the application waits for it to complete. Use the AnimationTime property to control the number of milliseconds the animation takes to execute.

The window's Open event is triggered before the animation starts, allowing any code that changes the size or layout of the window to complete before it is painted.

In a painter

❖ **To set the OpenAnimation property on a window:**

- Select a value from the OpenAnimation drop-down list on the General page of the window's Properties view

In scripts

The following example sets the OpenAnimation property of the w_about window to CenterAnimation!:

```
w_about.OpenAnimation = CenterAnimation!
```

See also

AnimationTime
CloseAnimation

OriginalSize

Applies to

Picture, PictureButton, PictureHyperLink, and Animation controls

Description

The OriginalSize property specifies whether the width and height of the picture are set to their original values.

For Animation controls, it specifies whether the width and height properties of the Animation control are set to the size of the AVI clip.

In the painter, if you use the mouse to resize the control or if you set the Width or Height properties on the Position tab, the OriginalSize property becomes disabled and the check box becomes unchecked.

Usage

In a painter

❖ **To set the picture to its original size:**

- Select the OriginalSize check box on the General page of the control's Properties view.

In scripts

The OriginalSize property takes a boolean value. The following line sets the OriginalSize property to false:

```
p_1.OriginalSize = FALSE
```

You should not try to change the width or height of a picture control when OriginalSize is set to true, because it can lead to unexpected behavior. In this example, the OriginalSize property is checked and set to false before the control is doubled in size:

```
integer li_width, li_height
li_width = p_1.width * 2
li_height = p_1.height * 2
parent.setredraw(false)
p_1.setredraw(false)

if p_1.originalsize then p_1.originalsize = FALSE
p_1.width = li_width
p_1.height = li_height
p_1.setredraw(TRUE)
parent.setredraw(TRUE)
```

The SetRedraw function must be called only when the image is very large. Before performing multiple resize operations of large JPEG images, set the name of the picture to an empty string temporarily to avoid unnecessary lengthy recompilations.

OriginLine

Applies to

grAxis objects in Graph controls

Description

The OriginLine property specifies the style of the line that represents the value zero for that axis in the graph. In the painter, the line style settings for an axis are disabled if the axis is not appropriate for the graph type.

Usage

In a painter

❖ **To select an origin line style:**

1 Display the Axis tab page from the graph's Properties view and select the desired axis from the Axis drop-down list.

2 Select the desired line style from the OriginLine list in the Line Style group.

In scripts

The OriginLine property takes a value of the LineStyle enumerated datatype. The following statement makes the Values axis origin line a dashed line:

```
gr_1.Values.OriginLine = Dash!
```

OverlapPercent

Applies to

Graph controls

Description

Overlap specifies how much 2D bar and column data markers in different series in a graph overlap. The number you specify is a percentage of the width of the data marker. This property is not applicable to all graph types.

Usage

In a painter

❖ **To set overlap for bar or column graphs:**

1 Select the desired 2D graph type from the GraphType list on the General page of the control's Properties view.

2 Use the OverlapPercent slide control to select the desired percentage of overlap.

In scripts

The OverlapPercent property takes an integer value. The following line sets the overlap to 10%:

```
gr_1.OverlapPercent = 10
```

PaperHeight

Applies to

RichTextEdit

Description

Sets the value for the display height of pages inside the control.

Usage

By default, the value you set for display height is multiplied by 1/1000 of an inch. An application user can change the display units at runtime when you enable the PopMenu property of the RichTextEdit control. This allows the application user to bring up the Rich Text Object dialog box and change the current units to 1/1000 of a centimeter. If the user switches the current units to centimeters, the values you set for PaperHeight and PaperWidth are multiplied by 2.54.

By default, the value you set for PaperHeight is used for printing as well as for screen display. When you set this value or the PaperWidth value, the default value in the Size drop-down list on the Print Specifications page of the Rich Text Object dialog box changes to `Customized`. Application users can modify the print specifications from the Rich Text Object dialog box at runtime, but only if you set the PopMenu property of the rich text object to `true`.

In scripts

The PaperHeight property takes a long value.

The following line sets the display height of a RichTextEdit to 11 inches.

```
rte_1.PaperHeight = 11000
```

PaperOrientation

Applies to

RichTextEdit

Description

Sets the orientation of document pages inside the control by switching the values for PaperHeight and PaperWidth.

Usage

In scripts

The PaperOrientation property takes an OrientationType enumerated value. Permitted values are PaperPortrait! and PaperLandscape!.

If you set the value to PaperPortrait! and the current PaperWidth is larger than the current PaperHeight, PowerBuilder switches these values so that the PaperHeight is larger. If you set the value to PaperPortrait! and the current PaperWidth is smaller than the current PaperHeight, PowerBuilder does not exchange these values.

If you set the value to PaperLandscape! and the current PaperHeight is larger than the current PaperWidth, PowerBuilder switches these values so that the PaperWidth is larger. If you set the value to PaperLandscape! and the current PaperHeight is smaller than the current PaperWidth, PowerBuilder does not exchange these values.

The following line sets the display orientation of a RichTextEdit to landscape.

```
rte_1.PaperOrientation = PaperLandscape!
```

By default, the value you set for PaperOrientation is used for printing as well as for screen display. The value in the Orientation drop-down list on the Print Specifications page of the Rich Text Object dialog box changes to the orientation value that you set for this property. Application users can modify print specifications from the Rich Text Object dialog box at runtime, but only if you set the PopMenu property of the rich text object to `true`.

PaperWidth

Applies to

RichTextEdit

Description

Sets the value for the display width of pages inside the control.

Usage

By default, the value you set for display width is multiplied by 1/1000 of an inch. An application user can change the display units at runtime when you enable the PopMenu property of the RichTextEdit control. This allows the application user to bring up the Rich Text Object dialog box and change the current units to 1/1000 of a centimeter. If the user switches the current units to centimeters, the values you set for PaperHeight and PaperWidth are multiplied by 2.54.

By default, the value you set for PaperWidth is used for printing as well as for screen display. When you set this value or the PaperHeight value, the default value in the Size drop-down list on the Print Specifications page of the Rich Text Object dialog box changes to `Customized`. Application users can modify the print specifications from the Rich Text Object dialog box at runtime, but only if you set the PopMenu property of the rich text object to `true`.

In scripts

The PaperWidth property takes a long value.

The following line sets the display width of a RichTextEdit to 8 inches.

```
rte_1.PaperWidth = 8000
```

Password

Applies to

SingleLineEdit controls

Description

The Password property specifies whether the control is a password field, in which characters the user types appear as asterisks (*). If Password is not enabled, the characters appear as the user types them.

Usage

In a painter

❖ **To make the control a password field:**

- Select the Password check box on the General page of the control's Properties view.

In scripts

The Password property takes a boolean value.

The following example sets the SingleLineEdit to a password field so that characters typed in appear as asterisks.

```
sle_1.Password = TRUE
```

PerpendicularText

Applies to

Tab controls

Description

When PerpendicularText is enabled, the tab labels are drawn perpendicular to the tab page, resulting in narrower tabs. When PerpendicularText is not enabled, text runs parallel to the edge of the tab page, resulting in wider tabs.

Usage

In a painter

❖ **To select a perpendicular orientation for tab text:**

- Select the Perpendicular Text check box on the General page of the control's Properties view.

In scripts

The PerpendicularText property takes a boolean value.

The following line specifies that tab labels are perpendicular to the tab page.

```
tab_1.PerpendicularText = TRUE
```

Perspective

Applies to

Graph controls Properties view

Description

Perspective controls the distance of a 3D graph from the front of the window. Perspective is not available for 2D graphs.

Usage

In a painter

❖ **To change the perspective of a 3D graph:**

1. Select the desired 3D graph type from the GraphType list on the General page of the graph control's Properties view.
2. Use the Perspective slide control to change the graph perspective.

In scripts

The Perspective property takes an integer value from 1 to 100. The larger the number, the greater the distance from the front of the window and the smaller the graph appears.

To set the distance (and size) of the graph, use a line like the following:

```
gr_1.Perspective = 25
```

PicturesAsFrame

Applies to

RichTextEdit controls

Description

When PicturesAsFrame is enabled, any bitmaps used in the control appear as empty frames. If this property is not enabled, graphics appear normally.

PicturesAsFrame can also be enabled by the user at runtime from the properties item on the pop-up menu.

Usage

In a painter

❖ **To display graphics as empty frames:**

- Select the PicturesAsFrame check box on the Document tab page of the control's Properties view.

In scripts

The PicturesAsFrame property takes a boolean value.

The following line specifies that graphics in a RichTextEdit appear as frames:

```
rte_1.PicturesAsFrame = TRUE
```

PictureHeight

Applies to

PictureListBox, DropDownPictureListBox, TreeView, Toolbar controls

Description

The PictureHeight property specifies in pixels the display height of all the pictures in the control. In a script, this property can be set only if there are no images in the PictureName property array. In the painter, you can change this value whether or not there are images in the Picture list.

Usage

In a painter

❖ **To set the picture height:**

- Select the desired value from the Height drop-down list on the Pictures tab page of the control's Properties view.

  The choices of 16 and 32 are standard pixel heights for icons. If you select Default, PowerBuilder uses the height of the first picture in the PictureName array as the height for all the pictures.

In scripts

The PictureHeight property takes an integer value. This value can be set only before the first call to the AddPicture function or after calling DeletePictures. If this value is set to 0, then the size of the first picture in the PictureName property array is used as the height for all the pictures.

The following line sets the height for a TreeView's pictures to 16 pixels.

```
tv_1.PictureHeight = 16
```

PictureIndex

Applies to

ListViewItem, TreeViewItem

Description

The PictureIndex property identifies pictures in the control's Picture list. For ListViewItems, the index identifies the large, small, and state picture associated with the item.

For TreeViewItems, the index identifies the picture displayed to the left of the item label. If the index is 0, no picture is displayed. You can set the PictureIndex property only for TreeViewItems with scripts, but you can add pictures to the control's Picture list in the painter.

Usage

In a painter

❖ **To associate pictures with a Listview item:**

1. Select the Large Picture tab page, Small Picture tab page, or State tab page from the ListView control's Properties view.
2. Do one of the following:
   - In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
   - Use the Browse button.

- Select one or more pictures from the Stock Pictures list.

3 Select the Items tab page from the ListView control's Properties view.

4 Use the row numbers from the Picture Name list to specify the Picture Index for each List View Item on the Items tab page.

❖ **To add pictures to a TreeView control's picture list:**

1 Select the Pictures tab page from the TreeView control's Properties view.

2 Do one of the following:

- In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
- Use the Browse button.
- Select one or more pictures from the Stock Pictures list.

You associate pictures in the TreeView control's picture list with TreeViewItems using scripts.

In scripts

This example illustrates how to get a ListViewItem object and change the value of PictureIndex:

```
listviewitem lvi
lv_1.GetItem(4, lvi)
lvi.PictureIndex = 2
lv_1.SetItem(4, lvi)
```

For more information about scripting ListView and TreeView controls, see "Using ListView Controls" and "Using TreeView Controls" in *Application Techniques*.

PictureMaskColor

Applies to

PictureListBox, DropDownPictureListBox, TreeView controls, TabPage user objects

Description

The PictureMaskColor property specifies the color in the picture that is transparent when the picture is displayed. You can change the mask color before adding each picture. Each image uses the mask color that was in effect when it was added.

Usage

In a painter

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

❖ **To set the picture mask color for ListBox and TreeView controls:**

1 Select the Pictures tab page of the control's Properties view.

2 Select the desired color from the Picture Mask Color drop-down list box.

❖ **To set the picture mask color for TabPage objects in a tab control:**

1 Select the desired TabPage object of the tab control.

2 Select the Picture tab page of the TabPage object's Properties view.

3 Select the desired color from the Picture Mask Color drop-down list box.

❖ **To set the picture mask color for a TabPage user object:**

1 Select the TabPage tab page of the user object's Properties view.

2 Select the desired color from the Picture Mask Color drop-down list box.

The mask color selected for the user object can be changed after it has been inserted into a tab control.

In scripts

The PictureMaskColor property takes a long value (-2 to 16,777,215) that specifies the numerical value of the mask color. The PictureMaskColor value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for a particular color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script. In scripts, this property is used when each picture is added and, therefore, can be changed between AddPicture calls.

The following example sets yellow as the mask color for pictures in a DropDownPictureListBox:

```
ddplb_1.PictureMaskColor = RGB(255, 255, 0)
```

PictureName

Applies to

Picture, PictureButton, and PictureHyperLink controls, UserObject used as tab page

Description

The PictureName property specifies the name of the file that contains the picture displayed in the control. For PictureButton controls, the picture specified by the PictureName property is the one that is displayed when the button is enabled.

The picture can be in the following formats:

- bitmap (*.BMP*)
- runlength encoded (*.RLE*)
- Windows metafile (*.WMF*)
- GIF (*.GIF*)
- JPEG (*.JPG* or *.JPEG*)

Usage

In a painter

❖ **To specify the picture for a Picture control and for the enabled state of a PictureButton:**

- Enter the name of the file in the PictureName field on the General page of the control's Properties view, or use the Browse button next to the PictureName field to select a file.

In scripts

The PictureName property takes a string value.

The following line selects a picture file for a PictureButton pb_1:

```
pb_1.PictureName = "c:\pictures\pb1.bmp"
```

PictureName[ ]

Applies to

TreeView, PictureListBox, DropDownPictureListBox, Toolbar controls

Description

The PictureName[ ] property specifies an indexed array of files containing the pictures used in the control. You can add pictures to the array in the painter, or use the AddPicture function at execution. However, adding or deleting pictures during execution does not update the PictureName property array.

The pictures can be in the following formats:

- bitmap (*.BMP*)
- GIF (*.GIF*)

- JPEG (*.JPG* or *.JPEG*)
- icon (*.ICO*)

Usage

In a painter

❖ **To add pictures to the PictureName array:**

1 Select the Pictures tab page from the control's Properties view.

2 Do one of the following:

- In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
- Use the Browse button.
- Select one or more pictures from the Stock Pictures list.

In scripts

The PictureName property array is populated at initialization and cannot be updated during execution.

The following example adds a picture to a TreeView control and associates it with a new TreeView item:

```
long ll_tvi
integer li_pic
li_pic = tv_1.AddPicture("c:\images\new.gif")
ll_tvi = tv_1.FindItem(RootTreeItem!, 0)
tv_1.InsertItemFirst(ll_tvi, "New", li_picture)
```

PictureOnRight

Applies to

Tab controls

Description

When the PictureOnRight property is enabled, the picture, if any, that is part of the tab label is to the right of the text. When PictureOnRight is not enabled, the picture is to the left of the tab label text.

Usage

In a painter

❖ **To specify pictures to the right of text on tab labels:**

- Select the Pictures on Right check box on the General page of the tab control's Properties view.

In scripts

The PictureOnRight property takes a boolean value.

The following line puts pictures to the right of the tab labels:

```
tab_1.PictureOnRight = TRUE
```

PictureWidth

Applies to

PictureListBox, DropDownPictureListBox, TreeView, Toolbar controls

Description

The PictureWidth property specifies in pixels the display width of all the pictures in the control. In a script, this property can be set only if there are no images in the PictureName property array. In the painter, you can change this value whether or not there are images in the Picture list.

Usage

In a painter

❖ **To set the picture width:**

- Select the desired value from the Width drop-down list on the Pictures tab page of the control's Properties view.

  The choices of 16 and 32 are standard pixel widths for icons. If you select Default, PowerBuilder uses the width of the first picture in the PictureName array as the width for all the pictures.

In scripts

The PictureWidth property takes an integer value. This value can be set only before the first call to the AddPicture function or after calling DeletePictures. If this value is set to 0, then the size of the first picture in the PictureName property array is used as the width for all the pictures.

The following line sets the width for a TreeView's pictures to 16 pixels:

```
tv_1.PictureWidth = 16
```

Pointer

Applies to

All controls

Description

The Pointer property specifies the pointer image displayed when the pointer is over a control.

Rich text objects

The RichTextEdit control supports only the pointers included in the list of stock pointers on the Other tab of the Properties view. The RichText DataWindow supports only the pointers listed on the Pointer tab page of the Rich Text Object property sheet.

If no pointer is specified, the default pointer is "IBeam!" when the rich text control is editable and "Arrow!" when the control is read-only.

Usage

In a painter

❖ **To specify a pointer:**

1 Select the Other tab page from the control's Properties view.

2 Do one of the following:

- Type the complete path and name of the file containing the pointer image in the Pointer text box.
- Use the Browse button.
- Click the down arrow to display a list of stock pointers and select a pointer from the list.

In scripts

The Pointer property takes a string containing either a file name or the name of one of the Pointer enumerated datatypes.

Both of the following lines set the I-beam as the pointer for CommandButton cb_1.

```
cb_1.Pointer = 'Beam!'
cb_1.Pointer = 'd:\archive\IBEAM.BMP'
```

PopMenu

Applies to

RichTextEdit controls

Description

When PopMenu is enabled, the user has access to a pop-up menu by clicking the right mouse button on the control. The pop-up menu allows the user to cut and paste, insert a file, and select formatting properties.

The pop-up menu can be disabled by the user at runtime from the Properties item on the menu itself.

Usage

In a painter

❖ **To enable the pop-up menu:**

- Select the Pop-up Menu check box on the Document tab page of the control's Properties view.

In scripts

The PopMenu property takes a boolean value.

The following line enables the pop-up menu for a RichTextEdit:

```
rte_1.PopMenu = TRUE
```

Port

Applies to

MLSynchronization, MLSync, and SyncParm objects

Description

Specifies the port used for the MobiLink synchronization server.

Usage

At design time, you can enter a value for Port on the MLServer tab of the Properties view for an MLSync object. At runtime, application users can enter a value for the port on the MLServer tab page of the default synchronization options window generated by the MobiLink wizard.

If the port is defined by subscriptions in the remote database, you do not need to set this property. Default ports are 2439 for TCP/IP connections, 80 for HTTP connections, and 443 for HTTPS connections.

In scripts

You can change the port name in script as follows:

```
mySync_1.port = 443
```

Position

Applies to

HProgressBar, VProgressBar, HScrollBar, VScrollBar, HTrackBar, VTrackBar controls

Description

Position specifies where the scroll box or slider appears when the scroll bar or track bar is first displayed at runtime. For a progress bar, Position specifies the value of the current position within the range of the control (set with the SetRange function). The control uses the range and the current position to determine the percentage of the progress bar to fill with the highlight color.

Usage

In a painter

❖ **To set the initial position of the progress indicator, scroll box, or slider:**

- Type a number that is between the values you have specified in MinPosition and MaxPosition.

In scripts

The Position property takes an integer value. It must be used in conjunction with MaxPosition and MinPosition.

For example, if a vertical scroll bar's minimum is 0 and maximum is 100, this statement positions the scroll box 80 percent of the way toward the bottom:

```
vsb_1.Position = 80
```

PowerTipText

Applies to

InkPicture, Picture, PictureButton, and PictureHyperlink controls, and UserObjects with tab pages

Description

Displays a PowerTip when the user moves a cursor over the control or over the tab area of the tab page.

Usage

In a painter

❖ **To set the PowerTip:**

1 In the Window painter, display the General page of the control's Properties view, or in the User Object painter, display the TabPage tab of the UserObject's Properties view.

2 Type a PowerTip in the box for the PowerTipText field.

In scripts

The PowerTipText property takes a string value.

The following line adds a PowerTip for tabpage_2 on tab control tab_1:

```
tab_1.tabpage_2.PowerTipText = "Cancel the operation"
```

This adds a PowerTip for a PictureButton control:

```
pb_1.PowerTipText = "This button opens a new form"
```

PowerTips

Applies to

Tab controls

Description

When the PowerTips property is enabled, any PowerTip text defined for a tab page is displayed as pop-up text when the mouse pointer pauses over the tab. PowerTips are useful if the tabs are labeled with pictures.

Usage

In a painter

❖ **To enable the display of PowerTip text:**

- Select the Power Tips check box on the General page of the Tab control's Properties view.

In scripts

The PowerTips property takes a boolean value. The following line allows display of PowerTips for each tab page.

```
tab_1.PowerTips = TRUE
```

PrimaryLine

Applies to

grAxis objects in Graph controls

Description

The PrimaryLine property specifies the line style for the primary line used for the axis itself. The line style settings for an axis are disabled if the axis is not appropriate for the graph type. Primary lines are not visible if the line style is set to transparent!.

Usage

In a painter

❖ **To set the primary line style for an axis:**

1. Display the Axis tab page from the graph's Properties view and select the desired axis from the Axis drop-down list.
2. Select the desired line style from the PrimaryLine drop-down list in the Line Style group.

In scripts

The PrimaryLine property takes a value of the LineStyle enumerated datatype. The following line sets the PrimaryLine property for the Values axis of a Graph to a dash:

```
gr_1.Values.PrimaryLine = Dash!
```

ProcessOption

Applies to

MLSynchronization and MLSync objects

Description

Specifies the direction for synchronization events. This property takes a value of the enumerated datatype SyncProcessType. Setting this property is equivalent to including the `-uo` dbmlsync option.

Usage

In a painter

On the Settings page of the object's Properties view, select the ProcessOption value that you want from the Process Option drop-down list. Values are UploadOnly!, DownloadOnly!, or Bidirectional!.

In scripts

You can modify ProcessOption values in script as follows:

```
mySync_1.ProcessOption = UploadOnly!
```

ProgressWindowName

Applies to

MLSynchronization and MLSync objects

Description

Specifies the name of a progress or status window used by the MobiLink synchronization application.

Usage

The MobiLink wizard generates a progress window that you can modify and use with all your MobiLink applications.

At design time, you can enter a value for ProgressWindowName on the Logging tab of the Properties view for an MLSync object. The progress window displays during synchronization only if the UseWindow property is set to true.

In scripts

You can change the progress window name in script as follows:

```
mySync_1.progresswindowname = &
    "w_mycustomized_progress_window"
```

Publication

Applies to

MLSynchronization and MLSync objects

Description

Specifies the publication or publications to be updated during a synchronization. Setting this property is equivalent to including the `-n` dbmlsync option.

Usage

This is a required property of the synchronization object. If the property value has not been set before a synchronize call, the synchronization fails and an error string is saved to the synchronization object's ErrorText property.

At design time, you can enter the Publication property value on the General tab of the Properties view for an MLSync object.

The default synchronization options window generated by the MobiLink wizard contains a Publications text box that is visible but disabled. At runtime, application users can view Publication values on the Subscriptions tab page of the default synchronization options window, but they cannot change those values unless you enable the text box field.

In scripts

You can modify Publication values in script as follows:

```
mlSync.Publication = "pubs1, pubs2"
```

RaggedRight

Applies to

Tab controls

Description

When RaggedRight is enabled, tab size is determined by the label text and the Fixed Width setting. If RaggedRight is not enabled, tabs are stretched so that they fill space along the edge of the control.

Usage

In a painter

❖ **To set the RaggedRight property:**

- Select the Ragged Right check box on the General page of the tab control's Properties view.

  When this check box is selected, the tabs are sized based on their label text and whether the Fixed Width check box is selected.

In scripts

The RaggedRight property takes a boolean value. The following line specifies that tabs are stretched to fill the edge of the control:

```
tab_1.RaggedRight = FALSE
```

RecognitionTimer

Applies to

InkEdit controls

Description

Specifies the time period in milliseconds between the last ink stroke and the start of text recognition.

Usage

In a painter

❖ **To specify that ink added to the InkEdit control is not converted to text:**

- Clear the InsertAsText check box on the Ink page in the Properties view.

In scripts

The RecognitionTimer property takes a long value. By default, ink is converted to text after two seconds (2000 milliseconds). If you want to give the user more time to enter text, increase the RecognitionTimer value.

This example sets the interval between the last stroke and the beginning of text recognition to one minute (60000 milliseconds):

```
ie_1.RecognitionTimer = 60000
```

Render3D

Applies to

Graph controls and Graph DataWindows

Description

You can check this option to render 3D graphs in the DirectX style.

Usage

In a painter

❖ **To change a graph to the DirectX style**

- Select the Render3D check box on the General page in the Properties view.

In scripts

The Render3D property takes a boolean value.

This statement sets a Graph control gr_1 to the DirectX 3D style.

```
gr_1.Render3D = TRUE
```

This statement sets a DataWindow in the Graph presentation style to the DirectX 3D style.

```
dw_1.Object.gr_1.Render3D = TRUE
```

Resizable

Applies to

DataWindow, OLE, and RichTextEdit controls and windows

Description

A resizable window or control has a thick border, and the user can use the mouse or the keyboard to resize it.

Usage

In a painter

❖ **To allow a user to resize a window or control:**

- Select the Resizeable check box on the General page of the window's or control's Properties view.

In scripts

The Resizable property takes a boolean value.

This statement makes the DataWindow control dw_1 resizable.

```
dw_1.Resizable = TRUE
```

ReturnCode

Applies to

SyncParm objects

Description — Stores the return code from a synchronization operation.

Usage — You can check the return code to determine whether synchronization was successful. The code returned is 0 for success, -1 for failure, and 100 if the synchronization was cancelled.

ReturnsVisible (obsolete)

Applies to — RichTextEdit controls

Description — When the ReturnsVisible property is enabled, characters for carriage returns in the text will display.

Obsolete property
This property is replaced by the ControlCharsVisible property.

RightMargin

Applies to — RichTextEdit controls

Description — The RightMargin property specifies the size in inches of the right margin on the printed page.

Usage

In a painter

❖ **To set the right margin:**

- Enter the desired size in inches in the Right Margin field of the Document tab page of the RichTextEdit control's property page.

In scripts

The RightMargin property takes a long value.

The following line sets the right margin of a RichTextEdit to 1 inch.

```
rte_1.RightMargin = 1
```

RightToLeft

Applies to

Application and Window objects, and CheckBox, DataWindow, DatePicker, DropDownListBox, DropDownPictureListBox, EditMask, GroupBox, InkEdit, ListBox, ListView, MonthCalendar, MultiLineEdit, PictureListBox, RadioButton, SingleLineEdit, StaticHyperLink, StaticText, and TreeView controls

Description

The RightToLeft property specifies that characters should be displayed in right-to-left order. The application must be running on an operating system that supports right-to-left display. Values are:

- TRUE – Characters display in right-to-left order.
- FALSE – Characters display in left-to-right order (default).

When you want to display Arabic or Hebrew text for message and buttons, set the RightToLeft property of the Application object to true. The characters of the message display from right to left. However, the button text continues to display in English unless you are running a localized version of PowerBuilder.

This property has no effect on other aspects of the control's layout. For ListView and TreeView controls, use the LayoutRTL property to display a mirror image of the standard layout.

For best results, set this property in the painter so that you can see its effect. Setting this property at runtime can have unexpected results.

For the TreeView DataWindow style, this property can be set in the development environment but it cannot be set at runtime.

Usage

In a painter

❖ **To set the RightToLeft property:**

- Select the RightToLeft check box on the General page in the control's Properties view.

In scripts

The RightToLeft property takes a boolean value.

The following line sets the RightToLeft property of a SingleLineEdit control to true:

```
sle_1.RightToLeft = TRUE
```

Rotation

Applies to

Graph controls

Description

The Rotation property specifies the rotation from left to right of 3D graphs. Rotation is disabled for 2D graphs.

Usage

In a painter

❖ **To set the rotation of the graph:**

1 Select a 3D graph type from the GraphType list on the General page of the graph control's Properties view.

2 Move the Rotation slider to change the graph's rotation.

In scripts

The Rotation property takes an integer value.

The following example rotates the graph 45 degrees to the left:

```
gr_1.Rotation = -45
```

RulerBar

Applies to

RichTextEdit controls

Description

When the RulerBar property is enabled, a ruler bar appears above the editing area of the control. The user can use it to set tabs and margins on the tab bar.

The ruler bar can also be enabled and disabled by the user at runtime from the Properties item on the pop-up menu, if the PopMenu property of the control has been set to true.

Usage

In a painter

❖ **To make the ruler bar visible:**

- Select the Ruler Bar check box on the Document tab page of the control's Properties view.

In scripts

The RulerBar property takes a boolean value.

The following line makes the ruler bar appear in a RichTextEdit:

```
rte_1.RulerBar = TRUE
```

RoundTo

Applies to

grAxis objects in Graph controls

Description

When the AutoScale property is enabled, the RoundTo and RoundToUnit properties specify how to round the end points and tick marks of an axis. Rounding affects axis labels, not graph data.

The RoundTo property specifies the value to which you want to round the axis values, in the units specified by the RoundToUnit property.

Usage

In a painter

❖ **To set the value to which to round axis values:**

1. Display the Axis tab page from the graph control's Properties view and select the desired axis from the Axis list.
2. Turn on autoscaling by checking the AutoScale check box.
3. Choose the datatype of the axis by selecting an option from the DataType drop-down list.
4. Enter a value in the RoundTo edit field.

In scripts

The RoundTo property takes a double value indicating the multiple to which you want to round axis tick marks.

The following example sets the datatype of the Values axis to date, sets the unit for rounding to months, and then sets the rounding value to six months:

```
gr_1.Values.DataType = AdtDate!
gr_1.Values.RoundToUnit = RndMonths!
gr_1.Values.RoundTo = 6
```

RoundToUnit

Applies to

grAxis objects in Graph controls

Description

When the AutoScale property is enabled, the RoundTo and RoundToUnit properties specify how to round the end points and tick marks of an axis. Rounding affects axis labels, not graph data.

The RoundToUnit property specifies the type of units that should be used for the rounding. The type of units that can be specified are based on the datatype of the axis. For example, for a date axis, you might round tick marks to the nearest five years or to every third month.

Usage

In a painter

❖ **To specify the type of unit to be used for rounding:**

1 Display the Axis tab page of the graph control's Properties view and select the desired axis from the Axis list.

2 Turn on autoscaling by checking the AutoScale check box.

3 Choose the datatype of the axis by selecting an option from the DataType drop-down list.

4 Choose the desired unit from the RoundToUnit drop-down list.

In scripts

The RoundToUnit property takes a value of the enumerated datatype grRoundToType. When you set this property in scripts, make sure the value is compatible with the datatype of the axis.

The following example sets the datatype of the Values axis to date and then sets the unit for rounding to months and the number of months to which to round:

```
gr_1.Values.DataType = AdtDate!
gr_1.Values.RoundToUnit = RndMonths!
gr_1.Values.RoundTo = 6
```

ScaleType

Applies to

grAxis objects in Graph controls

Description

The ScaleType property specifies the scale used for an axis. An axis can have linear or logarithmic scaling. The default is Linear. Other values are Log10 and LogE.

Usage

In a painter

❖ **To select the scale type for an axis:**

1 Display the Axis tab page of the graph control's Properties view and select the desired axis from the Axis list.

2 Select the desired type from the ScaleType drop-down list in the Scale group.

In scripts

The ScaleType property takes a value of the grScaleType enumerated datatype.

To set the scale type of the Values axis of gr_1 to log 10, use a line like the following:

```
gr_1.Values.ScaleType=Log10!
```

ScaleValue

Applies to

grAxis objects in Graph controls

Description

The ScaleValue property specifies the scale of values on the axis. You cannot set this property in the painter.

Usage

In scripts

The ScaleValue property takes a value of the grScaleValue enumerated datatype.

The following line sets the ScaleValue of the Values axis of a graph:

```
gr_1.Values.ScaleValue = Actual!
```

Scrolling

Applies to

ListView controls

Description

When Scrolling is enabled, the user can scroll vertically when some of the items in a ListView control are not visible. When Scrolling is not enabled, the user cannot scroll.

Usage

In a painter

❖ **To enable scrolling:**

- Select the Scrolling check box on the General page of the control's Properties view.

In scripts

The Scrolling property takes a boolean value. The following line enables scrolling when necessary in a ListView:

```
lv_1.Scrolling = TRUE
```

ScrollRate

Applies to

MonthCalendar controls

Description

Specifies the number of months the calendar scrolls when the user clicks a scroll button.

Usage

In a painter

❖ **To set the ScrollRate property:**

- Select a number from the ScrollRate spin control on the General page in the Properties view.

In scripts

The ScrollRate property takes an integer value. The default is 1, which means that the calendar scrolls by one month, regardless of how many months display. This example sets ScrollRate to 3:

```
mc_1.ScrollRate = 3
```

SecondaryLine

Applies to

grAxis objects in Graph controls

Description

The SecondaryLine property specifies the style of the lines used in the axis parallel to and opposite the primary axis in the graph.

Usage

In a painter

The line style settings for an axis are disabled in the painter if the axis is not appropriate for the graph type.

❖ **To set the secondary line style:**

1 Display the Axis tab from the graph control's Properties view and select the desired axis from the Axis list.

2 Select the desired line style from the SecondaryLine drop-down list in the Line Style group.

In scripts

The SecondaryLine property takes a value of the LineStyle enumerated datatype. The following example sets the SecondaryLine property of the Values axis of a graph to a dash:

```
gr_1.Values.SecondaryLine = Dash!
```

SelectedStartPos

Applies to

RichTextEdit controls

Description

The SelectedStartPos property specifies the starting position in a selected text string.

Usage

Typically, you use the SelectedStartPos property to set the starting position of a selected text string to the first letter of a word that is flagged by a supported ActiveX spell checking control.

See the chapter on implementing rich text in *Application Techniques* for more information about spell checking text in RichTextEdit controls.

In a painter

The SelectedStartPos property cannot be set in a design-time painter.

In scripts

The SelectedStartPos property takes a long value.

The following code in a ReplaceWord event for the Wintertree Software WSpell ActiveX control sets the starting position in the text string that is being spell checked to the offset position of a misspelled word. After setting the starting position, the SelectedTextLength setting causes the entire misspelled word to be highlighted, and the ReplaceText call replaces it with a word that the user selects in a WSpell dialog box.

```
string str
str = wspell.object.MisspelledWord
rte_1.SelectedStartPos = wspell.object.WordOffset
rte_1.SelectedTextLength  = Len(str)
rte_1.ReplaceText(wspell.object.ReplacementWord)
```

SelectedTab

Applies to

Tab controls

Description

The SelectedTab property specifies the index number of the selected tab page in the tab control. As the user selects tabs in the Tab control, the value of SelectedTab changes to reflect the currently selected tab.

Usage

In a painter

❖ **To specify the selected tab:**

- Enter a number in the Selected Tab field on the General page of the control's Properties view.

  The number should be in the range 1 to *N* where *N* is the number of tab pages in the tab control.

In scripts

The SelectedTab property takes an integer value. The default value is 1, and the integer must be in the range 1 to *N*, where *N* is the number of tab pages.

The following line sets the index number of the selected tab page in the Tab control tab_1 to 3:

```
tab_1.SelectedTab = 3
```

SelectedTextLength

Applies to

RichTextEdit controls

Description

The SelectedTextLength property specifies the length of text you want to highlight in a selected text string.

Usage

Typically you use this property to obtain the length of a misspelled word that is flagged after passing the selected text string to a supported ActiveX spell checking control.

See the chapter on implementing rich text in *Application Techniques* for more information about spell checking text in RichTextEdit controls.

In a painter

The SelectedTextLength property cannot be set in a design-time painter.

In scripts

The SelectedTextLength property takes a long value. The following code in a MixedCaseWord or ReplaceWord event for the Wintertree Software WSpell ActiveX control causes a word flagged by the control to be highlighted for its entire length, beginning with the word's offset position in the text string that you are spell checking:

```
string strword
strword = wspell.object.MisspelledWord
rte_1.SelectedStartPos = wspell.object.WordOffset
rte_1.SelectedTextLength = Len(strword)
```

Series

Applies to

Graph controls

Description

The Series property is used to define the properties of the Series axis in a graph. The Series axis is valid for 3D graphs only.

Usage

In a painter

❖ **To define the properties of the Series axis of a 3D graph:**

1 Select a 3D graph type from the GraphType list on the General page of the graph control's Properties view.

2 Display the Axis tab page of the graph control's Properties view and select Series in the Axis list.

All the properties of the Series axis can be set from the Axis tab page.

In scripts

The Series axis is an object of type grAxis within the Graph control. The Series object has its own properties for controlling the appearance of the axis.

The following line sets the scale type of the Series axis of gr_1 to log 10:

```
gr_1.Series.Scaletype = Log10!
```

SeriesSort

Applies to

Graph controls

Description

The SeriesSort property specifies how the series are sorted: ascending, descending, or unsorted.

Usage

In a painter

❖ **To specify how the series are sorted:**

- Select the desired sort type from the SeriesSort drop-down list on the General page of the graph control's Properties view.

In scripts

The datatype of the CategorySort property is the grSortType enumerated datatype, which has the values Ascending!, Descending!, Unsorted!, and UserDefinedSort!.

The following example specifies that the series should be unsorted:

```
gr_1.SeriesSort = Unsorted!
```

SetStep

Applies to

HProgressBar and VProgressBar controls

Description

A progress bar has a range and a current position. The SetStep property allows you to set the size of the increments by which the current position advances as progress is shown. The default value is 10.

Usage

In a painter

❖ **To set the increment size:**

- Use the spin control or enter an integer in the SetStep text box on the General page of the control's Properties view.

In a script

SetStep takes an integer value. In the following example, the range of the progress bar is set to 0 to 500, and the step value is set to 50:

```
hpb_1.setrange(0,500)
hpb_1.setstep = 50
```

ShadeBackEdge

Applies to

grAxis objects in Graph controls

Description

Specifies whether the back edge of an axis is shaded. Applies only to 3D graphs. The shade color is a property of the graph, not the axis.

Usage

In a painter

❖ **To shade the back edge of an axis in a 3D graph:**

1 Select a 3D graph type from the GraphType list on the General page of the graph control's Properties view.

2 Select a shade color from the ShadeColor list on the General page.

3 Display the Axis tab page of the graph control's Properties view and select the desired axis from the Axis list.

4 Select the ShadeBackEdge check box on the Axis tab page.

In scripts

The ShadeBackEdge property takes a boolean value. The following example selects the shade color for the graph and then specifies that the back edge of the Category axis in a 3D graph is shaded:

```
gr_1.ShadeColor = RGB(240,250,150)
gr_1.Category.ShadeBackEdge = TRUE
```

ShowList

Applies to

DropDownListBox, DropDownPictureListBox controls

Description

If the ShowList property is enabled, the option list is always displayed. If this property is not enabled, the list is displayed only when the user clicks on the control's down arrow.

Usage

In a painter

❖ **To specify that the option list should always be displayed:**

- Select the Always Show List check box on the General page of the control's Properties view.

In scripts

The ShowList property takes a boolean value. The following example specifies that the list of choices for the DropDownListBox should always be displayed:

```
ddlb_1.ShowList = TRUE
```

Note that the AllowEdit property must also be true when ShowList is true.

ShowHeader

Applies to

ListView controls

Description

When the ShowHeader property is enabled, column titles appear in the report view of a ListView control. When ShowHeader is not enabled, column titles do not appear in the report view.

To enable report view in a ListView control, you must write a script that establishes and populates columns. See "Using ListView controls" in *Application Techniques* for more information about enabling report view.

Usage

In a painter

❖ **To specify a header for report view:**

- Select the Show Header check box on the General page of the control's Properties view.

In scripts

The ShowHeader property takes a boolean value. The following line enables display of a header in report view.

```
lv_1.ShowHeader = TRUE
```

ShowPicture

Applies to

Tab controls

Description

When the ShowPicture property is enabled, the picture specified for each tab, if any, is displayed. When this property is not enabled, no pictures appear.

You can use ShowPicture with ShowText to display a picture and a text label, picture only, text label only, or no label at all.

Usage

In a painter

❖ **To show the pictures on the tab pages in the tab control:**

- Select the Show Pictures check box on the General page of the tab control's Properties view.

In scripts

The ShowPicture property takes a boolean value. The following line allows the picture, if any, to appear for each tab:

```
tab_1.ShowPicture = TRUE
```

ShowText

Applies to

Tab controls

Description

When the ShowText property is enabled, the text specified for each tab, if any, is displayed. When this property is not enabled, no text appears.

You can use ShowText with ShowPicture to display a picture and a text label, picture only, text label only, or no label at all.

Usage

In a painter

❖ **To show the text on the tab pages in the tab control:**

- Select the Show Text check box on the General page of the tab control's Properties view.

In scripts

The ShowText property takes a boolean value. The following line allows the text, if any, to appear for each tab:

```
tab_1.ShowText = TRUE
```

ShowUpDown

Applies to

DatePicker controls

Description

Specifies whether an up-down control is used to change the date in the DatePicker control.

The ShowUpDown property takes a boolean value. The default is false. When ShowUpDown is set to true, the drop-down arrow in the DatePicker control is replaced with an up-down control. The user can select individual elements in the date and/or time in the control (year, month, day, hours, minutes, and seconds) and increase or decrease them by one unit using the up or down arrows.

This property cannot be changed at runtime.

Usage

In a painter

❖ **To set the ShowUpDown property:**

- Select or clear the ShowUpDown check box on the General page in the Properties view

SmallPictureHeight

Applies to

ListView controls

Description

The SmallPictureHeight property specifies the display height of all the pictures in the Small Icon view of the ListView control. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the height of the first picture in the array as the height for all the pictures. The other choices in the painter, 16 and 32, are standard pixel heights for icons.

The type of picture used is determined by the value of the View property of the control.

Usage

In a painter

❖ **To set the small picture height:**

- Select a value from the Height drop-down list on the Small Picture tab page of the control's Properties view.

In scripts

The SmallPictureHeight property takes an integer value. This value can be set only before the first call to the AddSmallPicture function or after calling DeleteSmallPictures. If this value is set to 0, then the size of the first picture is used to set the size of small pictures.

The following line sets the height for small pictures in a ListView to 16 pixels:

```
lv_1.SmallPictureHeight = 16
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

SmallPictureMaskColor

Applies to

ListView controls

Description

The mask color is the color in the picture that is transparent when the picture is displayed.

Select the color to mask newly added user-defined bitmaps. In scripts, you can change the mask color before adding each picture. Each image uses the mask color that was in effect when it was added.

Usage

In a painter

❖ **To specify a picture mask color:**

- Select a color from the Picture Mask Color drop-down list on the Small Picture tab page of the control's Properties view.

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

In scripts

The SmallPictureMaskColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color. This property is used when each bitmap is added and, therefore, can be changed between AddSmallPicture calls.

The SmallPictureMaskColor value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for a particular color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the mask color for user-defined bitmaps in a ListView:

```
lv_1.SmallPictureMaskColor = RGB(255, 255, 0)
```

SmallPictureName[ ]

Applies to

ListView controls

Description

PowerBuilder stores ListView images in several indexed arrays of images. You can associate an image with a specific ListView item when you create a ListView in the painter or use the AddItem and InsertItem functions at execution time.

You identify a specific image by its index number. Because the same index number refers to both the large picture and the small picture for the item (depending on which view is selected), you need to make sure the images for each position in the array are compatible. The type of picture used in the control is determined by the value of the control's View property.

Usage

In a painter

❖ **To specify images for the Small Icon view**

1 Select the Small Picture tab page from the ListView control's Properties view.

2 Do one of the following:

- In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
- Use the Browse button.
- Select one or more pictures from the Stock Pictures list.

The order of the picture names specified here should match the picture name order used for the Large Icon view.

3 Use the row numbers from this Picture Name list to specify the Picture Index for each List View Item on the Items tab page.

In scripts

The SmallPictureName property takes a string value. You cannot use the SmallPictureName property to update the image list during execution. Use the AddSmallPicture function to add small pictures to a ListView control. For example:

```
lv_1.AddSmallPicture("c:\ArtGal\bmps\celtic.bmp")
```

When you add a small picture to a ListView control, it is given the next available picture index in the ListView.

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

SmallPictureWidth

Applies to

ListView controls

Description

The SmallPictureWidth property specifies the display width of all the pictures in the Small Icon view of the ListView control. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the width of the first picture in the array as the width for all the pictures. The other choices in the painter, 16 and 32, are standard pixel widths for icons.

Usage

In a painter

❖ **To set the small picture width:**

- Select a value from the Width drop-down list on the Small Picture tab page of the control's Properties view.

In scripts

The SmallPictureWidth property takes an integer value. This value can be set only before the first call to the AddSmallPicture function or after calling DeleteSmallPictures. If this value is set to 0, then the size of the first picture is used to set the size of small pictures.

The following line sets the width for small pictures in a ListView to 16 pixels.

```
lv_1.SmallPictureWidth = 16
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

Sorted

Applies to

DropDownListBox, DropDownPictureListBox, ListBox, PictureListBox controls

Description

Items in a list box can be sorted alphabetically. If the Sorted property is enabled, the items in the list box are sorted in ascending order. If this property is not enabled, the items in the list box are not sorted and are displayed in the order in which they were added.

Usage

In a painter

❖ **To enable automatic sorting:**

- Select the Sorted check box on the General page of the control's Properties view.

In scripts

The Sorted property takes a boolean value. The following line specifies that items in the ListBox lb_1 are sorted:

```
lb_1.Sorted = TRUE
```

SortType

Applies to

ListView, TreeView controls

Description

The SortType property specifies how items should be sorted. Items can be sorted alphabetically based on the item names or according to user-defined rules. If you specify a user-defined or unsorted sort type, define your sort criteria in the Sort event of the control.

In TreeView controls, each parent item's children form their own sorted list. For more information, see *Application Techniques*.

Usage

In a painter

❖ **To specify how items should be sorted:**

- Select the desired sort type from the Sort drop-down list on the General page of the control's Properties view.

In scripts

The SortType property takes a value of the grSortType enumerated datatype. The following line specifies Unsorted for the items in a ListView.

```
lv_1.SortType = Unsorted!
```

SpacesVisible (obsolete)

Applies to

RichTextEdit controls

Description

When SpacesVisible is enabled, spaces in the text are marked by a dot in the RichTextEdit control. If this property is not enabled, spaces appear as blanks only.

Obsolete property
This property is replaced by the ControlCharsVisible property.

Spacing

Applies to

Graph controls

Description

Spacing defines the gap (space) between data markers in a graph as a percent of the width of the markers. For example, in a bar graph, 100 is the width of one bar; 50 is half a bar.

Usage

In a painter

❖ **To change the spacing of data markers:**

- Select the desired spacing percentage using the Spacing slider on the General page of the graph control's Properties view.

In scripts

The Spacing property takes an integer value.

The following line specifies 120 percent of the bar width as the space between bars in a bar Graph:

```
gr_1.Spacing = 120
```

Spin

Applies to

EditMask controls

Description

The Spin property specifies whether the control is defined as a spin control that contains up and down arrows that the user can click to cycle through fixed values.

Usage

In a painter

❖ **To make an EditMask into a spin control:**

- Select the Spin Control check box on the Mask tab page of the control's Properties view.

In scripts

The Spin property takes a boolean value.

The following line specifies that the user can cycle through values in an EditMask:

```
em_1.Spin = TRUE
```

StatePictureHeight

Applies to

ListView, TreeView controls

Description

The StatePictureHeight property specifies the display height of all the state pictures. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the height of the first picture in the array as the height for all the pictures. The other choices in the painter, 16 and 32, are standard pixel heights for icons.

Usage

In a painter

❖ **To set the state picture height:**

- Select a value from the Height drop-down list on the State tab page of the control's Properties view.

In scripts

The StatePictureHeight property takes an integer value. This value can be set only before the first call to the AddStatePicture function or after calling DeleteStatePictures. If this value is set to 0, then the size of the first picture is used to set the size of state pictures.

The following line sets the height for state pictures in a ListView to 16 pixels.

```
lv_1.StatePictureHeight = 16
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

StatePictureMaskColor

Applies to

ListView, TreeView controls

Description

The mask color is the color in the picture that is transparent when the picture is displayed.

Select the color to mask newly added user-defined bitmaps. You can change the mask color before adding each picture. Each image uses the mask color that was in effect when it was added.

Usage

In a painter

❖ **To specify a picture mask color:**

- Select a color from the Picture Mask Color drop-down list on the State Picture tab page of the control's Properties view.

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

In scripts

The StatePictureMaskColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color. This property is used when each bitmap is added and, therefore, can be changed between AddStatePicture calls.

The StatePictureMaskColor value is a combination of values for the red, green, and blue components of the color. If you do not know the long value for a particular color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the mask color for user-defined bitmaps in a ListView:

```
lv_1.StatePictureMaskColor = RGB(255, 255, 0)
```

StatePictureName[ ]

Applies to

ListView, TreeView controls

Description

PowerBuilder stores ListView images in several indexed arrays of images. State pictures are displayed to the left of ListView items and their pictures, if they have them.

You can associate a state image with a ListView control only with scripts.

You identify a specific image by its index number.

Usage

In a painter

❖ **To specify State images:**

1 Select the State tab page from the ListView control's Properties view.

2 Do one of the following:

- In the rows provided in the Picture Name field, type the complete path and name of the files containing the desired pictures.
- Use the Browse button.
- Select one or more pictures from the Stock Pictures list.

3 Use the row numbers from this Picture Name list as the index number when setting the State picture index in scripts.

In scripts

The StatePictureName property takes a string value. You cannot use the StatePictureName property to update the image list during execution. Use the AddStatePicture function to add State pictures to a ListView control. For example:

```
integer index
index = lv_1.AddStatePicture("c:\ArtGal\ico\star.ico")
lv_1.StatePictureIndex = index
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

StatePictureWidth

Applies to

ListView, TreeView controls

Description

The StatePictureWidth property specifies the display width of all the state pictures. The size is specified in pixels.

If you choose the value (Default) in the painter, or set the value to 0, PowerBuilder uses the width of the first picture in the array as the width for all the pictures. The other choices in the painter, 16 and 32, are standard pixel widths for icons.

Usage

In a painter

❖ **To set the state picture width:**

- Select a value from the Width drop-down list on the State tab page of the control's Properties view.

In scripts

The StatePictureWidth property takes an integer value. This value can be set only before the first call to the AddStatePicture function or after calling DeleteStatePictures. If this value is set to 0, then the size of the first picture is used to set the size of state pictures.

The following line sets the width for state pictures in a ListView to 16 pixels:

```
lv_1.StatePictureWidth = 16
```

For more information about scripting ListView controls, see "Using ListView controls" in *Application Techniques*.

Status

Applies to

InkEdit, InkPicture controls

Description

A read-only property available at runtime that provides the current status of the control so that the user does not need to monitor the Stroke event to be sure that a stroke has been completed.

Usage

In scripts

The Status property for InkEdit controls takes a value of the InkEditStatus enumerated variable. Values are InkEditCollectingInk!, InkEditRecognizingInk!, and InkEditIdle!.

The Status property for InkPicture controls takes a value of the InkPicStatus enumerated variable. Values are CollectingInk! and Idle!.

This code in a button's Clicked event checks that the status of the InkEdit control is idle before setting the UseMouseForInput property to true:

```
IF ie_1.Status = InkEditIdle! THEN
   ie_1.UseMouseForInput = TRUE
ELSE
   MessageBox("Please try again later",  &
      "Text is being recognized.")
END IF
```

StdHeight

Applies to

HScrollBar

Description

If you enable the StdHeight property, the HScrollBar displays with the standard height for your system.

Usage

In a painter

❖ **To enable standard height:**

- Select the StdHeight check box on the General page of the control's Properties view.

In scripts

The StdHeight property takes a boolean value. At runtime, as long as StdHeight is true, setting the Height property has no effect. If you set the StdHeight property to true, the scroll bar displays with the standard height. If you set the StdHeight property to false, the scroll bar displays with the height specified in the Height property.

The following line specifies that height for an HScrollBar, instead of being standard, is set to the height specified in the Height property.

```
hsb_1.StdHeight = FALSE
```

StdWidth

Applies to

VScrollBar controls

Description

If you enable the StdWidth option, the VScrollBar displays with the standard width for your system.

Usage

In a painter

❖ **To enable standard width:**

- Select the StdWidth check box on the General page of the control's Properties view.

In scripts

The StdWidth property takes a boolean value. At runtime, as long as StdWidth is true, setting the Width property has no effect. If you set the StdWidth property to true, the scroll bar displays with the standard width. If you set the StdWidth property to false, the scroll bar displays with the width specified in the Width property.

The following line specifies that width for a VScrollBar, instead of being standard, is set to the width specified in the Width property.

```
vsb_1.StdWidth = FALSE
```

SyncRegistryKey

Applies to

MLSynchronization and MLSync objects

Description

Specifies the Windows registry key on the client computer where synchronization property values are stored.

Usage

At design time, you can enter a value for SyncRegistryKey on the Settings tab of the Properties view for an MLSync object.

In scripts

You can enter a synchronization registry key in script as follows:

```
mySync_1.syncregistrykey = &
    "Software\Sybase\PowerBuilder\17.0\myApp"
```

PowerBuilder prepends an "HKEY_CURRENT_USER\" prefix to the value you enter, and a "MobiLink" suffix.

TabBackColor

Applies to

TabPage objects and UserObjects when they are tab pages

Description

The TabBackColor property allows you to select the color of the tab on the tab page.

Windows XP
This property in not supported on Windows XP because the current XP theme controls the appearance of the tab on a tab page.

Usage

In a painter

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

❖ **To set the background color for the tab:**

1 Select the desired tab page on the tab control.

2 Select the TabPage tab in the Properties view.

3 Select the desired color in the TabBackColor drop-down list.

You can set the color of the body of the tab page on its General tab.

In scripts

The TabBackColor property takes a long value (-2 to 16,777,215) that specifies the numerical value of a color. The TabBackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets blue as the background color for a tab:

```
tab_1.tabpage_2.TabBackColor = RGB(0, 0, 255)
```

TabOrder

Applies to

Visible controls within a window.

Description

TabOrder specifies the order in which the control receives focus when the user tabs among controls within a window. Setting the TabOrder for a control to 0 means that the control cannot be tabbed to.

Usage

In a painter

❖ **To set tab order for controls within a window:**

1 Select Format>Tab Order from the menu bar.

Numbers indicating the tab order for each visible control are displayed in red on the window.

2 Select the number you want to change and type in a new number between 0 and 9999.

The actual value of the number does not matter; only the relative values among controls matter.

3 Select Format>Tab Order from the menu bar again to save the tab order.

In scripts

The TabOrder property takes an integer value between 0 and 9999. The value of 0 removes the control from the tab order.

The following example sets the tab order for three controls. The EditMask control is tabbed to after the ListView control and before the CommandButton:

```
lv_1.TabOrder = 10
em_1.TabOrder = 15
cb_1.TabOrder = 20
```

TabPosition

Applies to

Tab controls

Description

Tabs can appear on any side of the Tab control (top, bottom, left, right) or on opposite sides.

When you select two sides (for example, top and bottom), the selected tab divides the tabs so that tabs before it appear on one side and tabs after it appear on the opposite side. The selected tab itself appears on the first side of the pair.

Usage

In a painter

❖ **To set the position of tab pages in a tab control:**

- Select the desired type of position from the TabPosition drop-down list on the General page of the tab control's Properties view.

In scripts

The datatype of the TabPosition property is the TabPosition enumerated datatype.

The following example positions tabs on the top and bottom of the Tab control. Tabs before the selected tab and the selected tab itself are on top. Tabs after the selected tab are on the bottom.

```
tab_1.TabPosition = TabsOnTopAndBottom!
```

TabStop[ ]

Applies to

MultiLineEdit, EditMask, ListBox, PictureListBox controls

Description

The TabStop property array allows you to specify a repeating tab stop or tab stops at arbitrary positions. The tab stops are indicated by character positions.

If you specify one value, the tab stops are equally spaced using that value. If more than one tab stop is specified, tab stops are located in the character positions entered. The default is tab stops every 8 character positions.

Usage

In a painter

❖ **To specify tab stops:**

- Enter the character positions for each tab stop desired in the TabStop field on the General page of the control's Properties view.

In scripts

The TabStop[ ] property is a signed integer array containing the positions of the tab stops. The tab stops are in character positions.

The following lines define two tab stops at character positions 5 and 15.

```
lb_1.tabstop[1] = 5
lb_1.tabstop[2] = 15
```

TabTextColor

Applies to

TabPage objects and UserObjects when they are tab pages

Description

The TabTextColor property allows you to select the color for the tab's text.

Usage

In a painter

To add your own colors to the color drop-down list, select Design>Custom Colors before displaying the Properties view.

❖ **To change the tab text color:**

1 Select the desired tab page on the tab control.

2 Select the TabPage tab in the Properties view.

3 Select the desired color in the TabTextColor drop-down list.

In scripts

The TabTextColor property takes a long value (-2 to 16,777,215) that specifies the numerical value of a color. The TabTextColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets yellow as the text color for a tab.

```
tab_1.tabpage_2.TabTextColor = RGB(255, 255, 0)
```

TabsVisible (obsolete)

Applies to

RichTextEdit controls

Description

When the TabsVisible property is enabled, a text symbol appears for tabs in text in the RichTextEdit control.

Obsolete property
This property is replaced by the ControlCharsVisible property.

Tag

Applies to

All controls, user objects, and menus

Description

The Tag property can hold any text you want to associate with the control. It is up to you how you use that text.

Usage

In a painter

❖ **To specify a tag for a control:**

- Enter the desired text in the Tag field on the General page of the object's Properties view.

In scripts

The tag property takes a string value.

The following line uses the object's Tag property to set MicroHelp in an MDI frame (the code could be in a GetFocus event or, for a Menu object, the Selected event).

```
w_frame.SetMicroHelp(This.Tag)
```

Text

Applies to

Menus and controls that display text and DatePicker controls

Description

The Text property specifies the text displayed in the menu object or control.

If a Menu item has a shortcut key (for example, F1 or Alt+a), Text includes the shortcut key. If the Text property of a Menu item is a single dash (-), the item displays as a separator (a horizontal line the width of the menu), and all other properties for the item are ignored.

In DatePicker controls, the Text property is a read-only property that is equivalent to the Value property with the specified Format or CustomFormat applied. The Text property set for a DatePicker control must be capable of being converted to a DateTime value. This property cannot be set in the painter.

Usage

In a painter

❖ **To specify text to be displayed in a Menu item or control:**

- Enter the desired text in the Text field on the General page of the object's Properties view.

In scripts

The Text property takes a string value. The following line specifies that the text of a check box is `Male`:

```
cb_1.Text = "Male"
```

The following statements set the Format property to allow a custom format, then set the custom format, then return the text of the DatePicker control to the string variable *ls_text*:

```
string ls_text
dp_1.Format = dtfCustom!
dp_1.CustomFormat = "MMMM dd, yyyy"
ls_text = dp_1.text
```

TextCase

Applies to

EditMask, MultiLineEdit, SingleLineEdit controls

Description

The TextCase property lets you constrain the case of text entered by the user. The text can be displayed as the user types it, as all lowercase, or as all uppercase.

Usage

In a painter

❖ **To select the case used to display text entered by users:**

- Select the desired text case from the TextCase drop-down list on the General tag page in the control's Properties view.

In scripts

The TextCase property takes a value of the TextCase enumerated datatype. The following line sets the case for a MultiLineEdit to all uppercase:

```
mle_1.TextCase = Upper!
```

TextColor

Applies to

Controls and objects that display text

Description

The TextColor property specifies the color to be used for text in the control.

For the MonthCalendar control, TextColor is the color used to display text within a month.

Usage

This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.

In a painter

❖ **To set the text color for most controls:**

- Select the desired color from the TextColor drop-down list on the Font tab page for the control.

❖ **To set the text color for graph objects and MonthCalendar controls:**

- Select the desired color from the TextColor drop-down list on the General page of the Properties view.

❖ **To set the text color for text objects within graphs:**

1. Select the Text tab page from the graph's Properties view.
2. Select the desired text object from TextObject drop-down list.
3. Select a color from the TextColor drop-down list.

In scripts

The TextColor property is a long indicating the color to be used for the background for an object. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

In graphs, the TextColor property is a property of the graph object as well as of grDistAttr objects within the graph. For example, the following line sets text color for all the text objects in the Series Axis:

```
gr_1.Series.DispAttr.TextColor = RGB(0,0,255)
```

TextSize

Applies to

Controls that can display text

Description

The TextSize property specifies the point size of the text in the control.

Usage

This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.

In a painter

❖ **To set the size of all the text in a control:**

- Display the Font tab page of the control's Properties view and select the desired point size from the Size drop-down list, or select the control and then set the point size using the Font Size list box on the StyleBar.

❖ **To set the size of a text object in a graph control:**

1. Display the Text tab page of the graph control's Properties view and select the desired text object from the Text Object list.
2. Select the desired point size from the TextSize list.

❖ **To set the size of text in a menu:**

- Display the Appearance tab page of the top-level menu object's Properties view, select contemporarymenu! from the Menu Style list, and select the desired point size from the TextSize list.

  This property does not apply to menu items in the menu bar, which have a fixed size of 8 points.

In scripts

The TextSize property takes an integer value that indicates the point size. The following example sets the point size of a static text control:

```
st_1.TextSize = 12
```

This example sets the point size of the label of the Value axis of a graph control:

```
gr_1.Values.LabelDispAttr.TextSize = 12
```

ThreeState

Applies to

CheckBox controls

Description

The ThreeState property specifies whether or not the control can have three states. Typically, the state toggles between *selected* and *not selected*. For check boxes, if the ThreeState property has been enabled, the state of the control also toggles to a *third state*. A grayed-out mark is displayed for the *third state*.

Usage

In a painter

❖ **To allow the check box to have three states:**

- Select the ThreeState check box on the General page of the control's Properties view.

In scripts

The ThreeState property takes a boolean value. The following lines specify that a CheckBox can have three states and that it starts out in the third state:

```
cbx_1.ThreeState = TRUE
cbx_1.ThirdState = TRUE
```

ThirdState

Applies to

CheckBox controls

Description

The ThirdState property specifies whether the CheckBox is in the third state (neither selected nor unselected).

For a check box to be in the third state, the ThreeState property must also be enabled.

Usage

In a painter

❖ **To specify that a check box is in the third state:**

- Check both the ThreeState and the ThirdState check boxes on the General page of the control's Properties view.

In scripts

The ThirdState property takes a boolean value. The following lines specify that a CheckBox can have three states and that it starts out in the third state:

```
cbx_1.ThreeState = TRUE
cbx_1.ThirdState = TRUE
```

Title

Applies to

DataWindow controls, Graph controls, Windows

Description

The Title property specifies the title text of the control or window. In a window or DataWindow control, this value is displayed only if the TitleBar property is also enabled.

Usage

In a painter

❖ **To specify title text:**

- Type the title text in the Title field and select the TitleBar check box on the General page of the control's Properties view.

In scripts

The Title property takes a string value. The following lines set a title for a DataWindow control dw_1:

```
dw_1.TitleBar = TRUE
dw_1.Title = "Monthly Report"
```

TitleBackColor

Applies to

MonthCalendar controls

Description

The TitleBackColor property defines the color to be used for the background of the calendar's title.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select a color from the TitleBackColor drop-down list on the General page in the Properties view.

In scripts

The TitleBackColor property takes a long (-2 to 16,777,215) that specifies the numerical value of the background color of the month or months in a calendar. The TitleBackColor value is a combination of values for the red, green, and blue components of the color.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following example sets pale green as the background color for titles:

```
mc_1.TitleBackColor = RGB(128, 255, 128)
```

TitleBar

Applies to

DataWindow controls, Windows

Description

The TitleBar property specifies whether the DataWindow control or window displays a title bar. The user can move a window or DataWindow control only if it has a title bar.

If the window type is a main or MDI frame window with or without MicroHelp, the TitleBar property is always enabled. When the title bar is enabled, you can choose whether to include the control menu and the maximize and minimize boxes in the title bar.

Usage

In a painter

❖ **To display a title bar:**

- Select the TitleBar check box on the General page of the DataWindow control's or window's Properties view.

In scripts

For DataWindow controls, the TitleBar property can be modified in a script. It cannot be modified for Windows.

The TitleBar property takes a boolean value. The following line specifies that a title bar will appear in a DataWindow control dw_1:

```
dw_1.TitleBar = TRUE
```

TitleTextColor

Applies to

MonthCalendar controls

Description

The TitleTextColor property specifies the color used for text in the calendar's title.

Usage

This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.

In a painter

Select the desired color from the TitleTextColor drop-down list on the General tab page of the Properties view.

In scripts

The TitleTextColor property is a long indicating the color to be used for the title for a calendar. If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

For example, the following line sets the title text color for the control *mc_1*:

```
mc_1.TitleTextColor = RGB(0,0,255)
```

TodayCircle

Applies to

DatePicker, MonthCalendar controls

Description

Specifies whether a red circle or rectangle displays to highlight today's date on the calendar. The shape of the indicator depends on your operating system and display settings. If the TodaySection property is true, the indicator displays to its left.

Usage

In a painter

❖ **To set the TodayCircle property:**

- Select or clear the TodayCircle check box on the Calendar page in the Properties view for a DatePicker control or the General page in the Properties view for a MonthCalendar control.

In scripts

The TodayCircle property takes a boolean value. The default is true. This example turns the TodayCircle off in a DatePicker control:

```
dp_1.TodayCircle = false
```

TodaySection

Applies to

DatePicker, MonthCalendar controls

Description

Specifies whether the label "Today:" followed by the current date displays at the bottom of the calendar. If the TodayCircle property is true, a red rectangle displays to the left of the Today section.

Usage

In a painter

❖ **To set the TodaySection property:**

- Select or clear the TodaySection check box on the Calendar page in the Properties view for a DatePicker control or the General page in the Properties view for a MonthCalendar control.

In scripts

The TodaySection property takes a boolean value. The default is true. This example turns the TodaySection off in the calendar for a DatePicker control:

```
dp_1.TodaySection = false
```

ToolbarAlignment

Applies to

Windows

Description

In an MDI frame window, Alignment specifies where the toolbar displays.

Usage

In a painter

❖ **To specify the toolbar alignment:**

- Select the desired alignment type from the ToolbarAlignment drop-down list on the Toolbar tab page of the window's Properties view.

In scripts

The ToolbarAlignment property takes a value of the ToolbarAlignment enumerated datatype.

The following line specifies how the toolbar for window w_1 is aligned in the toolbar dock:

```
This.ToolbarAlignment = AlignAtRight!
```

ToolbarHeight

Applies to

Windows

Description

For MDI frame windows, the ToolbarHeight property specifies the height of the toolbar when it is a floating toolbar.

Usage

In a painter

❖ **To specify toolbar height:**

- Enter the desired height in the ToolbarHeight field on the Toolbar tab page in the window's Properties view.

In scripts

The ToolbarHeight property takes an integer value. The following line sets toolbar height for a window:

```
This.ToolbarHeight = 100
```

ToolbarVisible

Applies to

Windows

Description

For MDI frame windows, ToolbarVisible specifies whether the toolbar is displayed.

Usage

The ToolbarVisible property overrides the ToolbarItemVisible property for individual toolbar items.

In a painter

❖ **To make the toolbar visible:**

- Select the ToolbarVisible check box on the Toolbar tab page of the window's Properties view.

In scripts

The ToolbarVisible property takes a boolean value. The following line specifies that the toolbar for a window displays:

```
This.ToolbarVisible = TRUE
```

ToolbarWidth

Applies to

Windows

Description

For MDI frame windows, the ToolbarWidth property specifies the width of the toolbar when it is a floating toolbar.

Usage

In a painter

❖ **To specify toolbar width:**

- Enter the desired width in the ToolbarWidth field on the Toolbar tab page in the window's Properties view.

In scripts

The ToolbarWidth property takes an integer value. The following line sets toolbar width for a window:

```
This.ToolbarWidth = 500
```

ToolbarX

Applies to

Windows

Description

The ToolbarX property specifies the X coordinate in PowerBuilder units of the toolbar when it is a floating toolbar. The X coordinate is the distance from the left edge of the window or screen.

Usage

In a painter

❖ **To specify the X coordinate of the toolbar:**

- Enter the desired value, in PowerBuilder units, in the ToolbarX field on the Toolbar tab page of the window's Properties view.

In scripts

The ToolbarX property takes an integer value.

The following line specifies a distance of approximately 5 pixels from the left edge of the window for a toolbar:

```
This.ToolbarX = 20
```

ToolbarY

Applies to

Windows

Description

The ToolbarY property specifies the Y coordinate in PowerBuilder units of the toolbar when it is a floating toolbar. The Y coordinate is the distance from the top of the window or screen.

Usage

In a painter

❖ **To specify the Y coordinate of the toolbar:**

- Enter the desired value, in PowerBuilder units, in the ToolbarY field on the Toolbar tab page of the window's Properties view.

In scripts

The ToolbarY property takes an integer value. The following line specifies a distance of approximately 4 pixels from the top of the window for a toolbar:

```
This.ToolbarX = 15
```

ToolBar

Applies to

RichTextEdit controls

Description

When the ToolBar property is enabled, a toolbar for formatting text displays above the editing area of the RichTextEdit control. The toolbar includes bolding, italics, underscore, strikeout, alignment, spacing, superscript, subscript, tabs, display of text symbols such as paragraph returns, and text color. If the control is not wide enough, the tool bar is truncated.

The toolbar can also be enabled and disabled at runtime by the user from the Properties item on the pop-up menu, if the PopMenu property has been set to true.

Usage

In a painter

❖ **To display the toolbar in a RichTextEdit control:**

- Select the ToolBar check box on the Document tab page of the control's Property view.

In scripts

The ToolBar property takes a boolean value. The following line makes a toolbar display in a RichTextEdit:

```
rte_1.ToolBar = TRUE
```

TopMargin

Applies to

RichTextEdit controls

Description

The TopMargin property specifies the size in inches of the top margin on the printed page.

Usage

In a painter

❖ **To set the top margin:**

- Enter the desired size in inches in the Top Margin field of the Document tab page of the RichTextEdit control's Property view.

In scripts

The TopMargin property takes a long value. The following line sets the top margin of a RichTextEdit to 1 inch:

```
rte_1.TopMargin = 1
```

TrailingTextColor

Applies to

MonthCalendar controls

Description

The TrailingTextColor property specifies the color used for text for leading and trailing days in the calendar.

Usage

This property does not work on the Windows 7/8.1/10 operating system.

In a painter

Select the desired color from the TrailingTextColor drop-down list on the General tab page of the Properties view.

In scripts

The TrailingTextColor property is a long indicating the color to be used for leading and trailing days in the calendar. These are days in months that are partly displayed in the calendar. In a calendar showing a single month, they are the last few days of the preceding month and the next few days of the following month. In a calendar showing the three months July to September, the leading days are the last few days of June and the trailing days are the first few days of October. The default color is Disabled Text.

If you do not know the long value for the color, choose Design>Custom Colors to determine the red, green, and blue values and then call the RGB function to specify the color in a script.

The following line sets the trailing text color for the control mc_1 to "Inactive Title Bar":

```
mc_1.TrailingTextColor = 134217731
```

Transparency

Applies to

CheckBox, DropDownListBox, DropDownPictureListBox, EditMask, Graph, GroupBox, ListView, MultiLineEdit, PictureButton, RadioButton, RichTextEdit, SingleLineEdit, StaticHyperlink, Tab, TreeView, UserObject, and Window controls

Description

Specifies the transparency of a window.

Usage

The Transparency property takes an integer value in the range 0 to 100, where 0 means that the window is opaque and 100 that is completely transparent. This property is useful when you want a non-modal dialog box or window to remain visible but become semi-transparent when the user's focus has shifted to another area.

In MDI applications, sheet windows always have the same transparency as the frame window. The transparency setting of the sheet window is ignored.

In a painter

❖ **To make the window transparent:**

- Select or type a value in the Transparency spin control on the General page of the window's Properties view.

In scripts

Changes in the Transparency property take effect immediately.

The following statement sets Transparency to 25%:

```
w_popup.Transparency = 25
```

The following statement in the Moved event of a HTrackbar control sets the transparency based on the setting in the track bar:

```
w_popup.Transparency = scrollpos
```

Transparent

Applies to

Animation controls

Description

When the Transparent property is enabled, the animation control uses the same background color as its container, giving it a transparent appearance. You should also set the Border property to false.

Usage

In a painter

❖ **To make the control appear to be transparent:**

- Select the Transparent check box on the General page of the control's Properties view.

In scripts

The Transparent property takes a boolean value. The following line sets the Transparent property to true:

```
am_1.Transparent = TRUE
```

ULTrans

Applies to — SyncParm objects

Description — Not currently used.

Usage — Reserved for the connected transaction object to an UltraLite remote database.

Underline

Applies to — Controls that display text

Description — Underline is a property of text in a control.

Usage — This property does not work in MonthCalendar controls on the Windows 7/8.1/10 operating system.

In a painter

❖ **To underline all text items in a control:**

- Select the Underline check box on the Font tab page of the control's property page, or select the control and then click the U button on the StyleBar.

❖ **To underline a text object in a graph control:**

1. Display the Text tab page of the graph control's Properties view and select the desired text object from the Text Object list.
2. Select the Underline check box on the Text tab page.

In scripts

The Underline property takes a boolean value. The following example underlines the text in a StaticText control:

```
st_1.Underline = TRUE
```

This example underlines the label of the Value axis of a graph control:

```
gr_1.Values.LabelDispAttr.Underline = TRUE
```

UndoDepth (obsolete)

Applies to

RichTextEdit controls

Description

The UndoDepth property specifies the maximum number of editing changes that the Undo function will undo. Each time you call Undo, one more editing change is restored. The CanUndo function returns false when there are no more changes to undo.

Obsolete property
This property is ignored. The maximum undo depth is 50. This value cannot be changed at either design time or runtime.

UnitsPerColumn

Applies to

Windows and user objects

Description

UnitsPerColumn specifies the number of PowerBuilder units you want to scroll right or left when the user clicks the left or right arrow in the horizontal scroll bar in a window or user object. The default is 0 (1/100 of the width of the window or user object). PowerBuilder controls horizontal scrolling automatically when Units Per Column is 0.

PowerBuilder multiplies Units Per Column by Columns Per Page to determine the number of PowerBuilder units to scroll the window horizontally when the user clicks in the scroll bar.

For information on calculating ColumnsPerPage and UnitsPerColumn, see Scrolling in windows and user objects on page 591.

Usage note
To control the vertical scroll bar in a window or user object, use the UnitsPerLine and LinesPerPage properties.

Usage

In a painter

❖ **To specify the UnitsPerColumn property:**

- Enter the desired number of PowerBuilder units in the UnitsPerColumn field on the Scroll tab page of the window's Properties view.

In scripts

The UnitsPerColumn property takes an integer value.

The following statement sets Units Per Column to 12, which is appropriate for a content width of 1650:

```
This.UnitsPerColumn = 12
```

UnitsPerLine

Applies to

Windows, user objects

Description

UnitsPerLine specifies the number of PowerBuilder units you want to scroll up or down when the user clicks the up or down arrow in the vertical scroll bar in a window or user object. The default is 0 (1/100 of the window or user object height). When UnitsPerLine is 0, PowerBuilder controls vertical scrolling automatically.

PowerBuilder multiplies UnitsPerPage by UnitsPerLine to determine the number of PowerBuilder units to scroll the window or user object vertically when the user clicks in the scroll bar.

For information on calculating LinesPerPage and UnitsPerLine, see Scrolling in windows and user objects on page 591.

Usage note
To control horizontal scrolling in a window or user object, use the UnitsPerColumn and ColumnsPerPage properties.

Usage

In a painter

❖ **To set the UnitsPerLine property:**

- Enter the desired number of PowerBuilder units in the UnitsPerLine field on the Scroll tab page of the window's Properties view.

In scripts

The UnitsPerLine property takes an integer value.

The following statement sets UnitsPerLine to 17, which is appropriate for a content length of 2400:

```
lb_1.UnitsPerLine = 17
```

UseCodeTable

Applies to

EditMask controls

Description

When an EditMask control has been defined as a spin control (that is, a control with up and down arrows the user clicks to cycle through predefined values), a code table can be used to validate data.

The UseCodeTable property specifies whether the control uses a code table to validate data.

Usage

In a painter

❖ **To specify use of a code table for an EditMask control:**

1 Select the Spin Control and Code Table check boxes on the Mask tab page of the control's Properties view.

An area appears on the lower half of the tab page where you can enter values for the code table.

2 Specify Display Values and their corresponding Data Values.

Use the Insert button to insert items within this list.

In scripts

The UseCodeTable property takes a boolean value. This example specifies that the EditMask control should use its code table to validate data:

```
em_1.UseCodeTable = TRUE
```

You can specify the contents of the code table in scripts by using the DisplayData property. Enter the Display values and their corresponding Data values as a text string, with the Display and Data pairs separated by tabs and the pairs separated by slashes. For example:

```
em_1.DisplayData = "Black   1/White   2/Red   3"
```

UseLogFile

Applies to

MLSynchronization and MLSync objects

Description

Specifies whether to log synchronization processing information.

Usage

At design time, you can select the Save to Log File check box on the Logging tab of the Properties view for an MLSync object. You must also supply a log file name or set the LogFileName property if you want to save synchronization information.

At runtime, application users can select or clear the Use Log File check box on the Settings tab page of the default synchronization options window generated by the MobiLink wizard.

In scripts

You can change the UseLogFile value in script as follows:

```
mySync_1.UseLogFile = true
```

UseMouseForInput

Applies to

InkEdit, InkPicture controls

Description

Specifies whether the mouse can be used for input on a Tablet PC. This property has no effect on other computers.

Usage

In a painter

❖ **To specify that ink can be added using a mouse:**

- Select the UseMouseForInput check box on the Ink page in the Properties view.

In scripts

The UseMouseForInput property takes a boolean value. Do not change this property at runtime while the control is collecting or recognizing ink.

This code in a button's Clicked event checks that the status of the InkEdit control is idle before setting the UseMouseForInput property to true:

```
IF ie_1.Status = InkEditIdle! THEN
   ie_1.UseMouseForInput = TRUE
ELSE
   MessageBox("Please try again later",  &
      "Text is being recognized.")
END IF
```

UseWindow

Applies to

MLSynchronization and MLSync objects

Description

Specifies whether to display a progress window during synchronization.

Usage

At design time, you can select the Use Progress Window check box on the Logging tab of the Properties view for an MLSync object. You must also supply a progress window name or set the ProgressWindowName property if you want to display a progress window.

At runtime, application users can select the Display if Available radio button option on the Settings tab page of the default synchronization options window generated by the MobiLink wizard. Otherwise users can select the Do Not Display radio button option to prevent a progress window from displaying.

In scripts

You can change the UseWindow value in script as follows:

```
mySync_1.UseWindow = true
```

Value

Applies to

DatePicker controls

Description

Specifies the date/time value assigned to the control. The date part of the Value property uses the format for short dates specified in the regional settings in the Windows control panel on the local computer. The time part of the Value property uses the time format specified in regional settings.

The Value defaults to the current date and time.

Usage

In the painter

❖ **To set the Value property:**

- Select a date from the Value drop-down calendar on the General page of the Properties view or type a date and optional time into the Value box.

In scripts

The Value property takes a DateTime value. The value you assign must be capable of being interpreted as a DateTime value. You can use the DateValue and TimeValue properties to extract the date and time parts of the Value property.

This example sets the Value property to midday on July 1, 2005. The display format depends on the value of the Format and CustomFormat properties and the regional settings for date and time on the local computer:

```
dp_1.Value =
DateTime(Date("2005/07/01"),Time("12:00:00"))
```

View

Applies to

ListView controls

Description

A ListView has four ways to display its items:

- **Large icon view** Items are arranged from left to right and the user can move items around when drag and drop is enabled. Each item's picture is taken from the large picture list, and the item label is below the picture.
- **Small icon view** Same as large icon view except each item's picture is taken from the small picture list, and the item label is to the right of the picture.
- **List view** Items are arranged from top to bottom. Each item's picture is taken from the small picture array.
- **Report view** Items are arranged from top to bottom with one or more columns of information for each item. You must write a script to set up the columns.

Usage

In a painter

❖ **To select the view type:**

- Select the desired view type from the View drop-down list on the General page of the control's Properties view.

In scripts

The View property takes a value of the ListViewView enumerated datatype.

The following line specifies that small pictures appear for the items in the ListBox:

```
lv_1.View = ListViewSmallIcon!
```

Visible

Applies to

Controls, windows, user objects, menus

Description

The Visible property specifies whether the object, window object, or Menu object is visible.

Usage

In a painter

❖ **To set the Visible property:**

- Select the Visible check box on the General page of the object's Properties view.

In scripts

The Visible property takes a boolean value. The following line specifies that MultiLineEdit mle_1 is visible:

```
mle_1.Visible = TRUE
```

You can use the Show and Hide functions to change the visibility of an object.

Usage note
You cannot use the Visible property or the Show or Hide functions to show or hide an MDI sheet or a drop-down or cascading menu or any menu that has an MDI frame window as its parent window.

VScrollBar

Applies to

DataWindow, DropDownListBox, DropDownPictureListBox, EditMask, InkEdit, ListBox, MultiLineEdit, PictureListBox, RichTextEdit controls, windows, user objects

Description

When the VScrollBar property is enabled, PowerBuilder adds a vertical scroll bar to the right of a window or other control when the contents of the object are outside the borders.

Usage

In a painter

❖ **To allow display of a vertical scroll bar:**

- Select the VScrollBar check box on the General or Scroll tab page of the object's Properties view.

In scripts

The VScrollBar property is a boolean value.

This example displays a vertical scroll bar in a DataWindow control:

```
dw_1.VScrollBar = TRUE
```

This property cannot be set at runtime for EditMask controls.

VTextAlign

Applies to

PictureButton controls

Description

The HTextAlign property specifies how the text label for the PictureButton control is aligned in relation to the picture.

Usage

In a painter

❖ **To set the vertical alignment of text:**

- Select the desired alignment from the VTextAlign drop-down list on the General tab of the control's Properties view.

In scripts

The VTextAlign property takes a value of the VTextAlign enumerated datatype.

The following example specifies bottom alignment for text in a PictureButton:

```
pb_1.VTextAlign = Bottom!
```

WeekNumbers

Applies to

DatePicker, MonthCalendar controls

Description

Specifies whether a number displays to the left of each row of dates to indicate the number of the week in the year. For example, January 1 falls in the first week in the year, so the number 1 would display to the left of its row of data if this property is set to true.

Usage

In a painter

❖ **To set the WeekNumbers property:**

- Select or clear the WeekNumbers check box on the Calendar page in the Properties view for DatePicker controls or the General page in the Properties view for MonthCalendar controls.

In scripts

The WeekNumbers property takes a boolean value. The default is false. These statements turn the WeekNumbers property on for a DatePicker control and a MonthCalendar control:

```
dp_1.WeekNumbers = true
mc_1.WeekNumbers = true
```

Weight

Applies to

Controls that can display text

Description

The Weight property specifies the stroke weight of the text in the control.

Usage

In a painter

❖ **To set the stroke weight of all text in a control:**

- Display the Font tab page of the control's Properties view and select the Bold check box, or select the control and then click the B button on the StyleBar.

❖ **To set the stroke weight of a text object in a graph control:**

1. Display the Text tab page of the graph control's Properties view and select the desired text object from the Text Object list box.
2. Select the Bold check box.

In scripts

The Weight property takes an integer value. 400 indicates a normal weight and 700 indicates a bold weight. The following example sets the text labels of the tab pages of a tab control to bold:

```
tab_1.Weight = 700
```

Width

Applies to

Visible controls, windows

Description

The Width property specifies the width of a control or window in PowerBuilder units.

Usage

In a painter

❖ **To set the width of a control or window**

- Enter the desired width in the Width edit box on the Other tab page of the object's Properties view, or select the control or window and resize it with your cursor.

In scripts

The Width property takes an integer value specifying the width of an object in PowerBuilder units. The following example sets the width of a DataWindow control dw_1:

```
dw_1.Width = 750
```

It is illegal to resize a minimized or maximized sheet or frame. Changing the Width or Height property for a minimized or maximized window is not supported.

WindowDockOptions

Applies to

Child Windows

Description

WindowDockOptions are for child windows to specify how they can be opened:

- **WindowDockOptionAll!**
- **WindowDockOptionTabbedDocumentOnly!**
- **WindowDockOptionDockedOnly!**
- **WindowDockOptionFloatOnly!**
- **WindowDockOptionTabbedDocumentAndDockedOnly!**
- **WindowDockOptionTabbedDocumentAndFloatOnly!**
- **WindowDockOptionDockedAndFloatOnly!**

Usage

In a painter

❖ **To set the window docking:**

- Select the desired state from the WindowDockOptions drop-down list on the Docking tab page of the window's Properties view.

In scripts

You cannot specify the initial state of the window before it has been opened. You can change its display state afterwards while the window is open.

The WindowDockOptions property takes a value of the WindowDockOptions enumerated datatype. The following line sets the dock options for the current window:

```
This.WindowDockOptions = windowdockoptionfloatonly!
```

WindowDockState

Applies to

Windows

Description

The WindowDockState property specifies how the MDI windows are first displayed. The state can be:

- **Docked** The sheet is open and fixed in position relative to the Window object. The docked state is the default.
- **Floating** Users can move a floating sheet around or even outside the Window object.
- **TabbedDocument** Sheets that appear tabbed in the same area of the Window.
- **TabbedWindow** Docked windows that occupy the same area of the Window are in a tabbed group. The tabs are at the bottom.

Usage

In a painter

❖ **To set the window state:**

- Select the desired option from the WindowDockState drop-down list on the Position tab page of the window's Properties view.

In scripts

You cannot specify the initial state of the window before it has been opened. You can change its display state afterwards while the window is open.

The WindowDockState property takes a value of the WindowDockState enumerated datatype. The following line sets the Docked state for the current window:

```
This.WindowDockState = WindowDockStateDocked!
```

WindowObject

Applies to

MLSynchronization and MLSync objects

Description

Specifies an instance of a synchronization progress window. The class name of WindowObject must match the value of the ProgressWindowName property.

Usage

In scripts

You can set the WindowObject value in script as follows:

```
mySync_1.WindowObject = w_myProgressWindow
```

WindowState

Applies to

Windows

Description

The WindowState property specifies how the window is first displayed. The state can be:

- **Maximized** Enlarge the window to its maximum size.
- **Minimized** Shrink the window to an icon.
- **Normal (Default)** Display the window as it is defined in the painter.

Usage

In a painter

❖ **To set the window state:**

- Select the desired state from the WindowState drop-down list on the Position tab page of the window's Properties view.

In scripts

You cannot specify the initial state of the window before it has been opened. You can change its display state afterwards while the window is open.

The WindowState property takes a value of the WindowState enumerated datatype. The following line sets the Maximized state for the current window:

```
This.WindowState = Maximized!
```

WindowType

Applies to

Windows

Description

The value of this property specifies the type of window.

Table 3-3: Window types

Child	A window that is dependent on a main window and can exist only within the main (parent) window.
Main	A standalone overlapped window that can be independent of all other windows.
MDI	An MDI frame without a MicroHelp status bar.
MDIHelp	An MDI frame with a MicroHelp status bar.
MDIDock	A dockable MDI frame with a MicroHelp status bar.
MDIDockHelp	A dockable MDI frame with a MicroHelp status bar.
Popup	A window that usually displays in response to an event within a window, but can exist outside of the window and, in some cases, after the window that opened it is closed.
Response	A window that displays to obtain information from the user and cannot lose focus or be closed until the user responds.

Usage

In a painter

❖ **To specify the window type:**

- Select the desired type from the WindowType drop-down list on the General page of the window's Properties view.

In scripts

You cannot change a window's WindowType property dynamically at runtime.

WordWrap

Applies to

RichTextEdit controls

Description

WordWrap determines how a rich text control displays large blocks of text that do not contain spaces or other word-breaking characters (tab characters or end-of-line markers, but not hyphens). If the last word in a block of text is too large to fit on a line when WordWrap is enabled, the rich text control splits the word and displays the nonfitting characters on the following line.

When WordWrap is disabled, users cannot enter characters (other than word-breaking characters) beyond the right margin, and must move the cursor to a new line to continue entering text. If a document is inserted when WordWrap is disabled, and the document conatins a block of text too large to fit on a line, the rich text control hides the nonfitting characters, even when the text eventually breaks to a new line because of a space or other word-breaking character. .

WordWrap can be enabled or disabled by the user at runtime from the Properties item on the pop-up menu when the PopMenu property is enabled. If characters from an inserted text are hidden because WordWrap is disabled, and the user subsequently enables WordWrap, the hidden characters will be displayed on the next line of the rich text control. If the same text is inserted when WordWrap is enabled, and the user subsequently disables WordWrap, the rich text control hides previously visible characters that had wrapped to the next line.

Usage

In a painter

❖ **To enable word wrap:**

- Select the WordWrap check box on the Document tab page of the control's Properties view.

In scripts

The WordWrap property takes a boolean value.

The following line enables word wrapping for a RichTextEdit control:

```
rte_1.WordWrap = TRUE
```

X

Applies to

Controls, windows

Description

The X property specifies the X coordinate of an object or control in PowerBuilder units.

The X coordinate is the distance from the left edge of the window or custom user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To set the X coordinate:**

- Enter the desired X coordinate, in PowerBuilder units, in the X field of the Other tab page of the object's Properties view, or drag and drop the control to the desired location.

In scripts

The X property takes an integer value. The following line sets the distance from the left edge of a window for a DataWindow control dw_1:

```
dw_1.X = 215
```

You can also set the X and Y properties of a control using the Move function.

It is illegal to move a maximized sheet or frame. Changing the X or Y property for a maximized window is ignored.

Y

Applies to

Controls, windows

Description

The Y property specifies the Y coordinate of an object or control in PowerBuilder units. The Y coordinate is the distance from the top of the window or user object. If the object is a main window or custom user object, the distance is relative to the screen. If it is not a main window, the distance is relative to the parent window unless it is opened in an MDI frame window, in which case the distance is relative to the MDI frame.

Usage

In a painter

❖ **To set the Y coordinate:**

- Enter the desired Y coordinate, in PowerBuilder units, in the Y field of the Other tab page of the object's Properties view, or drag and drop the control to the desired location.

In scripts

The Y property takes an integer value. The following line sets the distance from the top of the window for a DataWindow control dw_1:

```
dw_1.Y = 215
```

You can also set the X and Y properties of a control using the Move function.

It is illegal to move a maximized sheet or frame. Changing the X or Y property for a maximized window is ignored.

CHAPTER 4

About Display Formats and Scrolling

About this chapter

This chapter describes how to use specific display formats with PowerBuilder controls and provides information about scrolling in PowerBuilder windows and user objects.

Contents

Using colors with display formats

Applies to

Display formats

Description

You can define a color for each display format section by specifying a color keyword before the format. The color keyword is the name of the color, or a number that is the color's RGB value, enclosed in square brackets. For example:

```
[RED]m/d/yy
[255]m/d/yy
```

The following table lists the named color keywords.

Table 4-1: Named color keywords

[BLACK]	[MAGENTA]
[BLUE]	[RED]
[CYAN]	[WHITE]
[GREEN]	[YELLOW]

The formula for combining primary color values into a number is:

256 * 256 * *blue* + 256 * *green* + *red* = *number*

where the amount of each primary color is specified as a value from 0 to 255. For example, to specify cyan, substitute 255 for blue, 255 for green, and 0 for red. The result is 16776960.

The following table lists the blue, green, and red values you can use in the formula to specify other colors.

Table 4-2: Values used to specify colors

Blue	Green	Red	Number	Color
0	0	255	255	Red
0	255	0	65280	Green
0	128	255	32768	Dark Green
255	0	0	16711680	Blue
0	255	255	65535	Yellow
0	128	128	328896	Brown
255	255	0	16776960	Cyan
192	192	192	12632256	Light Gray

Using date display formats

Applies to

Display formats

Description

A date display format can have two sections. The first section is required. The second section is optional and specifies how to represent NULLs:

date-format;null-format

The following table shows characters that have special meaning in date display formats.

Table 4-3: Special characters in date display formats

Character	Meaning	Example
d	Day number with no leading zero	9
dd	Day number with leading zero if appropriate	09
ddd	Day name abbreviation	Mon
dddd	Day name	Monday
m	Month number with no leading zero	6
mm	Month number with leading zero if appropriate	06
mmm	Month name abbreviation	Jun
mmmm	Month name	June
yy	Two-digit year	97
yyyy	Four-digit year	1997

Colons, slashes, and spaces display as entered in the mask.

Usage

If users specify a two-digit year in a DataWindow object, PowerBuilder assumes the date is the 20th century if the year is greater than or equal to 50. If the year is less than 50, PowerBuilder assumes the 21st century.

For example:

- 1/1/85 is interpreted as January 1, 1985
- 1/1/40 is interpreted as January 1, 2040

Examples

The following table shows how the date Friday, Jan. 30, 2003, displays when different format masks are applied.

Table 4-4: Date format examples

Format	Displays
`[red]m/d/yy`	`1/30/03` (in red)
`d-mmm-yy`	`30-Jan-03`
`dd-mmmm`	`30-January`
`mmm-yy`	`Jan-03`
`dddd, mmm d, yyyy`	`Friday, Jan 30, 2003`

Using number display formats

Applies to

Display formats

Description

A number display format can have up to four sections. Only the first is required.

```
Positive-format;negative-format;zero-format;null-format
```

The following table shows characters that have special meaning in number display formats.

Table 4-5: Special characters in number display formats

Character	Meaning
#	A number
0	A required number; a number will display for every 0 in the mask

Dollar signs, percent signs, decimal points, parentheses, and spaces display as entered in the mask.

These keywords tell PowerBuilder to determine an appropriate format based on system settings:

- [General]
- [Currency]

Examples

The following table shows how the values 5, -5, and .5 display when different format masks are applied.

Table 4-6: Number format examples

Sample format	5	-5	.5
[General]	5	-5	0.5
0	5	-5	1
0.00	5.00	-5.00	0.50
#,##0	5	-5	1
#,##0.00	5.00	-5.00	0.50
$#,##0;($#,##0)	$5	($5)	$1
$#,##0;-$#,##0	$5	-$5	$1
$#,##0;[RED]($#,##0)	$5	($5)	$1
$#,##0.00;($#,##0.00)	$5.00	($5.00)	$0.50
$#,##0.00;[RED]($#,##0.00)	$5.00	($5.00)	$0.50
0%	500%	-500%	50%
0.00%	500.00%	-500.00%	50.00%
0.00E+00	5.00E+00	-5.00E+00	5.00E-01

Using string display formats

Applies to

Display formats

Description

A string display format can have two sections. The first section is required. The second section is optional and specifies how to represent NULLs.

string-format;null-format

The following table shows characters that have special meaning in string display formats.

Table 4-7: Special characters in string display formats

Character	Meaning
@	A character

All other characters (including spaces) display as entered in the mask.

Examples

This format mask:

```
[red](@@@) @@@-@@@@
```

displays the string 800YESCELT in red as:

```
(800) YES-CELT
```

Using time display formats

Applies to

Display formats

Description

A time display format can have two sections. The first section is required and contains the format for times. The second section is optional and specifies how to represent NULLs.

```
time-format;null-format
```

The following table shows characters that have special meaning in time display formats.

Table 4-8: Special characters in time display formats

Character	Meaning
h	Hour with no leading zero (for example, 1).
hh	Hour with leading zero if appropriate (for example, 01).
m	Minute with no leading zero (must follow h or hh).
mm	Minute with leading zero if appropriate (must follow h or hh).
s	Second with no leading zero (must follow m or mm).
ss	Second with leading zero (must follow m or mm).
ffffff	Microseconds with no leading zeros. You can enter one to six f's; each f represents a fraction of a second (must follow s or ss).
AM/PM	Two-character, upper-case abbreviation (AM or PM as appropriate).
am/pm	Two-character, lower-case abbreviation (am or pm as appropriate).
A/P	One-character, upper-case abbreviation (A or P as appropriate).
a/p	One-character, lower-case abbreviation (a or p as appropriate).

Colons, slashes, and spaces display as entered in the mask.

The keyword [Time] tells PowerBuilder to use the time format specified in the Microsoft Windows control panel.

Usage

24-hour format is the default. Times display in 24-hour format unless you specify AM/PM, am/pm, A/P, or a/p.

Examples

The following table shows how the time 9:45:33:234567 PM displays when different format masks are applied.

Table 4-9: Time format examples

Format	Displays
h:mm AM/PM	9:45 PM
hh:mm A/P	09:45 P
h:mm:ss am/pm	9:45:33 pm
h:mm	21:45
h:mm:ss	21:45:33
h:mm:ss:f	21:45:33:2
h:mm:ss:fff	21:45:33:234
h:mm:ss:ffffff	21:45:33:234567
m/d/yy h:mm	1/30/03 21:45

Scrolling in windows and user objects

For scrolling purposes, PowerBuilder divides the window content into 100 lines and 100 columns. Lines, columns, and pages for scrolling do not correlate with any visible aspect of the window (such as the viewable area).

A *line* or a *column* is the amount scrolled by clicking a scroll bar arrow. There are 100 lines and 100 columns in the control being scrolled, regardless of the area occupied by the content of the window or user object. To get to the end of the scroll bar, the user can click 100 times on the scroll bar arrow.

A *page* is the amount scrolled by clicking in the scroll bar, not on the scroll bar arrows.

Vertical versus horizontal scrolling

The procedures in the following sections define vertical scrolling, determined by the UnitsPerLine and LinesPerPage properties, but the same formulas apply to horizontal scrolling, determined by the UnitsPerColumn and ColumnsPerPage properties.

Relating scrolling to height of content

If you want the bottommost content in the window to be visible when the user reaches the end of the scroll bar, you need to set the value of the control's UnitsPerLine property so that 100 lines cover the entire contents.

❖ **To determine the value for UnitsPerLine:**

1 Resize the window to include all the contents.

2 Look at the value of the Height option on the Position tab page of the window's Properties view.

The height is shown in PowerBuilder units (PBUs).

3 Divide 75% of the value of the Height option by 100 to get the number of PBUs each line should include:

```
UnitsPerLine = height * .75 / 100
```

Using 75% of the total height in this calculation keeps the end of the contents visible when the scroll bar reaches the end, instead of scrolling just out of sight.

Relating scrolling to page size

When the user clicks in the scroll bar, not on the scroll bar arrows, the control scrolls by a page. The page size is calculated using this formula:

```
pagesize = LinesPerPage * UnitsPerLine
```

Therefore, you can use the LinesPerPage property in conjunction with the UnitsPerLine property to set the page size for scrolling.

❖ **To determine the value for LinesPerPage**

1 Calculate the value of the UnitsPerLine property, as shown above.

2 Size the window to its desired final size.

3 Determine the height of the visible window area by looking at the value of the Height option on the Position tab page of the window's Properties view.

4 Decide how much of the window you want to have scroll every time the scroll bar is clicked. This will give you the page size in PBUs.

For example, if the visible window area height is 1200 PBUs and you want 1/4 of the window to scroll with each click, then the page size should be 300 PBUs.

5 Calculate the value of the LinesPerPage property.

For example:

```
LinesPerPage = 300 / UnitsPerLine
```

Scrolling using a fixed number of clicks

Alternatively, if you want to let the user get to the bottom of the content in a given number of clicks, regardless of the visible window area, set LinesPerPage using this formula:

```
LinesPerPage = 100 / number of clicks
```

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Y