



**Real's HowTo PDF version**

# Table of Contents

<b>Real's HowTo PDF version (november 2019).....</b>	<b>1</b>
<b>Common problems.....</b>	<b>3</b>
<u>pb-common.....</u>	3
<u>Table not visible.....</u>	3
<u>"numeric or value error" error when calling Oracle stored proc.....</u>	3
<u>Editmask cause application crash.....</u>	4
<u>Watcom C++ editor is not executed.....</u>	4
<u>Remove extra padding blanks in char column.....</u>	4
<u>GPF when scrolling via the WheelMouse on a datawindow.....</u>	5
<u>ODBC drivers installed but not visible.....</u>	5
<u>GPF when debugging with PB7.....</u>	5
<u>Avoid Blank sheets when printing reports.....</u>	6
<u>GPF on DW SetFullState (PB6.0).....</u>	6
<u>Memory usage on SQL Server connection.....</u>	6
<u>Accelerator key do not display in W2K.....</u>	6
<u>"Access or violation" error using timestamp with Oracle.....</u>	7
<u>No commit on GPF.....</u>	7
<u>Hide SqlAnywhere.....</u>	7
<u>Fix LibraryImport problem.....</u>	8
<u>Show the underline for keyboard shortcut.....</u>	8
<u>Allow user:password in URL.....</u>	8
<u>Initialization file is not writable. Cannot continue.....</u>	9
<b>Database.....</b>	<b>11</b>
<u>pb-database.....</u>	11
<u>Suppress ODBC logon window.....</u>	11
<u>List available ODBC datasources.....</u>	11
<u>How to get the current DBMS, Database or user through ODBC.....</u>	12
<u>Create a table from PowerScript.....</u>	13
<u>Prevent creation of PB special table.....</u>	13
<u>Get time information from the DBMS through ODBC.....</u>	13
<u>Enable database tracing.....</u>	14
<b>Datawindow.....</b>	<b>15</b>
<u>pb-datawindow.....</u>	15
<u>Load an array with the names of every column in a dw.....</u>	15
<u>Have a different color for newly inserted row.....</u>	15
<u>Set a Date to NULL.....</u>	15
<u>Create dynamically a DataWindow.....</u>	16
<u>Get the page count of a report.....</u>	16
<u>Make the ENTER key act as TAB key.....</u>	17
<u>Display only distinct rows.....</u>	17

# Table of Contents

## Datawindow

<u>Extract the display value of DDDW</u> .....	17
<u>Do simple validation with the Datawindow properties</u> .....	18
<u>Set a default value for DDDW</u> .....	19
<u>Delete current selection in DDDW</u> .....	19
<u>Alternate row color</u> .....	20
<u>Pass the content of a dw to another window</u> .....	20
<u>Detect which button is clicked in a dw (PB6)</u> .....	20
<u>Detect if a row is new</u> .....	21
<u>Set cursor at end of text in a dwcolumn</u> .....	21
<u>Make a static text visible on the last page only</u> .....	21
<u>Print to a file</u> .....	21
<u>Make a dw with a title unmoveable</u> .....	22
<u>Trap the click on down Arrow on DDLB</u> .....	22
<u>Scroll 2 datawindows in sync</u> .....	22
<u>Get the current filter</u> .....	23
<u>Disable a dw button</u> .....	23
<u>Prevent vertical scrolling via the mouse wheel or keyboard</u> .....	23
<u>Cut and Paste datawindow content</u> .....	24
<u>Open a DDDW via Powerscript</u> .....	24
<u>Change the presentation style</u> .....	24
<u>Autoselect an EditMask</u> .....	25
<u>Sort an array</u> .....	25
<u>Get the count of rows selected</u> .....	26
<u>Get data from Excel via the clipboard</u> .....	26
<u>Make a datawindow static text flash</u> .....	27
<u>Suppress datawindow built-in popup</u> .....	27
<u>Print a datawindow into a PDF</u> .....	27
<u>View datawindow content while in DEBUG mode</u> .....	28

## PFC

<u>pb-pfc</u> .....	30
<u>Disconnect an inactive application</u> .....	30
<u>Display compile timestamp in the w_about window</u> .....	30
<u>Use the w_master pfc Save event</u> .....	31
<u>Control the Cut&amp;Paste items in m_master</u> .....	32
<u>Have PFC Help accessible through the Toolbar</u> .....	34
<u>Have a FileExists function with wildcard</u> .....	34
<u>Specify objects to be save during a pfc Save event</u> .....	34
<u>Detect PFC version</u> .....	35

# Table of Contents

<b>Powerscript.....</b>	<b>36</b>
pb-powerscript.....	36
Get the PBL name.....	36
Time the execution.....	36
Rename an object.....	36
Get the numeric value for a color.....	37
Retrieve an environment variable.....	38
Common XP environment variables.....	38
Blank an array.....	39
Make a reference to an instance variable from a Menu.....	40
Make a reference to a Menu from a frame or a sheet.....	40
Get the Frame from a sheet.....	40
Get the remainder of a decimal.....	40
Detect PB version.....	41
Make a window stay "on top".....	41
Attach a menu in a response window.....	41
Navigate through a Tab control with keyboard.....	41
Remove the rtf information from the rte content.....	42
Capitalize a name.....	42
Convert an hex string to its decimal equivalent.....	42
Have a window close itself after a delay.....	43
Have a "mouse-over" on a component.....	43
Convert a number to hexadecimal.....	44
Evaluate an expression.....	44
Detect if running in PB or executable.....	45
Change text on a button to bold.....	45
Sort an array.....	46
Get data from the Internet.....	46
Transform a KeyCode! to a String.....	48
Use the undocumented INDIRECT keyword.....	49
Encrypt/Decrypt a string.....	56
Check if an event is supported by an object.....	57
Check if an object definition exists.....	58
Have a button with many lines.....	59
Display Powerbuilder's easter egg.....	59
Close a window with ESC.....	59
Terminate an another application.....	60
Get the current function/function name.....	60
Send email with attachment (MAPI).....	61
Use a Java class (PB9).....	61
Know more about the Java support in PB9.....	63
Check if the current user belongs a specific Windows group/role.....	63
Enable/disable a component from its name.....	64

# Table of Contents

## **Powerscript**

<u>Encode and display character using hex code</u> .....	64
<u>Use the clipboard</u> .....	65
<u>Overload a global function</u> .....	66
<u>Override system functions</u> .....	67
<u>Unicode String versus ANSI String (PB10)</u> .....	68
<u>Remove accented letters from a String</u> .....	69
<u>Get the execution path of the current application</u> .....	71
<u>Get the current directory</u> .....	72
<u>Change directory</u> .....	72
<u>Create or remove a directory</u> .....	73
<u>Rename a file</u> .....	73
<u>Generate a unique filename</u> .....	73
<u>Determine the TEMP directory designated for temporary files</u> .....	74
<u>Check if a file is open from another process</u> .....	75
<u>Read big file (&gt;32765 bytes)</u> .....	76
<u>Set File Attribute</u> .....	76
<u>Browse for a folder using the standard Win dialog</u> .....	77
<u>Retrieve a file timestamp</u> .....	78
<u>Convert a short pathname to a long</u> .....	80
<u>Truncate a long path with ellipses</u> .....	81
<u>Validate a date</u> .....	81
<u>Obtain the day of the year</u> .....	82
<u>Obtain the week of the year</u> .....	82
<u>Set the system date/time</u> .....	82
<u>Obtain the last day of a month</u> .....	83
<u>Get the time and date from a server</u> .....	84

## **WinAPI/Registry**.....88

<u>pb-winapiregistry</u> .....	88
<u>Get the network domain name</u> .....	88
<u>Use a Java object from PB</u> .....	91
<u>Use WSH-VBScript functionalities</u> .....	91
<u>Check if a program is running</u> .....	94
<u>Get information for the current user from ActiveDirectory</u> .....	95
<u>Run a program and wait</u> .....	95
<u>Create a shortcut</u> .....	96
<u>Use XML</u> .....	97
<u>Encode/decode URL's parameters</u> .....	97
<u>Display PDF into a Window</u> .....	98
<u>Write a Window Log Event</u> .....	99
<u>Print a Word document</u> .....	99
<u>Remove HTML tags to keep only text (with IE)</u> .....	100

# Table of Contents

## WinAPI/Registry

<u>Get CPU or other hardware/software infos (with WMI)</u> .....	100
<u>Use MQSeries</u> .....	105
<u>Terminate a Windows application</u> .....	110
<u>Get the current user email using Active Directory</u> .....	112
<u>Get a value from a JSON String</u> .....	113
<u>Get a value from a REST service</u> .....	114
<u>Start the default browser</u> .....	115
<u>Upload a file using FTP</u> .....	116
<u>Start a dial-up connection</u> .....	117
<u>Start/Run services</u> .....	117
<u>Display default email window to type and send email</u> .....	117
<u>Use Internet Explorer</u> .....	117
<u>Use MQSeries</u> .....	122
<u>Use RunDll32 utility</u> .....	127
<u>Get a list of printers installed</u> .....	129
<u>Use file association to start an application</u> .....	129
<u>Shutdown from application</u> .....	131
<u>Start the screen saver</u> .....	134
<u>Get the CDROM drive letter</u> .....	134
<u>Get the user name and the computer name</u> .....	135
<u>Drive mapping to UNC notation</u> .....	136
<u>Make a program sleep</u> .....	137
<u>Call HtmlHelp</u> .....	137
<u>Maximize a frame</u> .....	138
<u>Make a window popup "on top"</u> .....	138
<u>Make the "hourglass" cursor stay</u> .....	138
<u>Move a window without a titlebar</u> .....	139
<u>Convert ANSI to Oem character set</u> .....	139
<u>Send keystroke to a control</u> .....	140
<u>Return an ERRORLEVEL to a BAT file</u> .....	140
<u>Change screen resolution</u> .....	141
<u>Determine what is the decimal or hundred separator at runtime</u> .....	143
<u>Move the mouse cursor</u> .....	144
<u>Disable the MouseWheel ZOOM function</u> .....	145
<u>Set the system date/time</u> .....	145
<u>Remove the Close item in the system menu</u> .....	146
<u>Flash a Window Title bar</u> .....	147
<u>Detect if an application is already running</u> .....	148
<u>Retrieve error from calling a Win API</u> .....	148
<u>Use the default MAPI profile</u> .....	150
<u>Set and retrieve the executable version</u> .....	150
<u>Get the application PID</u> .....	152

# Table of Contents

## WinAPI/Registry

<u>Create and use a C DLL (MSVC6)</u> .....	153
<u>Use Windows Resources from a DLL</u> .....	155
<u>Get the IP address</u> .....	156
<u>Use Windows Debug API</u> .....	158
<u>Animate a Window</u> .....	159
<u>Use Microsoft Crypto API</u> .....	160
<u>Scroll a Window</u> .....	164
<u>Get OS version</u> .....	165
<u>Retrieve the Regional Setting w/o using the Registry</u> .....	166
<u>Make a window unmoveable</u> .....	168
<u>Retrieve window handle by its title</u> .....	169
<u>Have a transparent window</u> .....	169
<u>Bypass Window Error popup message</u> .....	170
<u>Get hard disk serial number</u> .....	170
<u>Make a selected item in a ListView visible</u> .....	171
<u>Get the CPU speed</u> .....	172
<u>Disable the logon and logoff button</u> .....	172
<u>Disable the screensaver</u> .....	173
<u>List Windows processes or services</u> .....	173
<u>Terminate a Windows process</u> .....	174
<u>Terminate a Windows application</u> .....	175
<u>Control CD audio tray, play MP3 file</u> .....	177
<u>Detect if user is using Terminal Server</u> .....	179
<u>Detect if a network is present</u> .....	179
<u>Detect network and/or internet connectivity state</u> .....	180
<u>Detect Caps Lock state</u> .....	181
<u>Set or Unset the keyboard numlock and read its state</u> .....	181
<u>Convert Win API calls to PB10</u> .....	182
<u>Hide a window from the Windows task list</u> .....	183
<u>Get the number of items in a Treeview</u> .....	184
<u>Detect if the current user is a member of the Administrator's group</u> .....	184
<u>Resize a window because XP window title is bigger</u> .....	184
<u>Convert the datatypes from Microsoft Win API to PowerBuilder</u> .....	185
<u>Allow user:password in URL</u> .....	185
<u>Get the time and date from a server</u> .....	186
<u>Set the wallpaper</u> .....	190
<u>Scan Window titles</u> .....	191
<u>Get an UUID (Universally Unique Identifiers)</u> .....	192
<u>Encode/decode base64 string</u> .....	193
<u>Get the executable name</u> .....	197
<u>Get Network card description and Mac address</u> .....	197
<u>Turn on/off the monitor</u> .....	198

## Table of Contents

### WinAPI/Registry

<u>Detect if running in 64bit OS</u> .....	199
<u>Display a window on the second monitor</u> .....	199

# Real's HowTo PDF version (november 2019).

This is the PDF version of the Real's HowTo Web site ( <https://www.rgagnon.com/howto.html> ). For up-to-date content, please refer to the Web site. There are 4 files : Real's Java , Real's Javascript, Real's Powerbuilder and Real's VBS and Misc Prog HowTo. Please don't make PDF versions available on the internet (it's ok in intranet). From the PDF, you can't run the examples and the links to other How-to's are not working.

---

If you feel that effort has been useful to you, perhaps you will consider giving something back? You can make a donation through PayPal at <https://www.paypal.com> , make you donation to [real@rgagnon.com](mailto:real@rgagnon.com) Contributions via PayPal are accepted in any amount using a credit card or checking account.

(Donations of **any size** gladly accepted)

---

This site is covered by the Creative Commons by-nc-sa license : [Click for more details about the license](#)  
You can share, adapt and reuse but not for commercial reason. See the [FAQ](#)

## Real's Howto copyright notice ( [real@rgagnon.com](mailto:real@rgagnon.com) )

Redistribution and use in source and binary forms,  
with or without modification, are permitted provided  
that the following conditions is met:

- \* the source code is used in a development project

Redistributions of source code or site content  
(even partially) in any publications (electronic or paper)  
is forbidden without permission.

## DISCLAIMER

THIS CONTENT IS PROVIDED BY Real Gagnon "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL Real Gagnon BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# **POWERBUILDER HOW-TO**

# Common problems

---

## pb-common

---

### Table not visible

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0106.html>

When you create a table, your userid is appended to the table name, for example if your user id/logon id is dba and you create MyTable, then this table is referenced as dba . MyTable. If you connect to it as dba, then you can just say MyTable (if you don't supply an owner prefix it assumes you are the owner), but any other user must refer to it as dba . MyTable.

This also affects how the table names display in the database painter and how the SELECT statements are painted for datawindows. If you connect to it as dba in the development environment, then the table name shows up in the painter as MyTable, and any datawindow created has a select statement that starts as SELECT FROM "MyTable". If another user, Foo, tries to execute the app and retrieve the DW, the DBMS prefixes his user name to the table name (because none was specified in the SELECT statement) and looks for Foo . MyTable, and thus cannot find it.

*The easiest way to deal with this is to connect in the development environment with a user name other than the table owner whenever you create or modify a DW to force the inclusion of the owner prefix in the generated SELECT statement.*

In this case the SELECT statement created will include the table owner prefix (i.e. dba . MyTable).

### "numeric or value error" error when calling Oracle stored proc

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0107.html>

If you are passing variables to Oracle stored procedure by reference, you need to assign memory first to receive the result. For example, if you are expecting to return a string of length 10, try initializing the variable.

```
[Oracle stored proc]
create procedure p_ret ( as_name OUT varchar2 ) as
begin
  as_name := 'SAM' ;
end ;

[PB local extenal function declaration]
```

```
SUBROUTINE p_ret ( REF string as_name ) RPCFUNC ;  
  
[PB calling the stored proc]  
ls_name = space(10)  
SQLCA.p_ret(ls_name)
```

## Editmask cause application crash

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0108.html>

It's probably because the Editmask is too small, simply increase the size of the field. This can happen when you design the screen in high resolution mode and you execute the application in a lower resolution. So make sure that Editmask field is wide enough for all resolutions.

## Watcom C++ editor is not executed

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0109.html>

The problem :

Trying to invoke C++ editor in the user object painter by clicking on the c++ button, but there is nothing happen: no message, no IDE.

The possible solution:

Check the environment variables related to Watcom ( eg. LIB, INCLUDE and the one about the EDITOR and make sure that there are no spaces in them ( ex. change c:\Program Files\Sybase\... to c:\Progra~1\Sybase\... ).

NOTE: Watcom C++ is no longer included with PB since v7.0 .

## Remove extra padding blanks in char column

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0110.html>

If a char(5) column contains only "a", some drivers are returning "a " and some "a" with the extra spaces removed.

For example, the trimming is done with SQLServer or Access but not for DB/2 or Ingres DBMS. Padding blanks in Char column are Ok according to the ANSI SQL standard it may be easier to assume that extra right spaces are trimmed.

You can let PB do the job for you by adding this parameter in the *PBODB60.INI* file (in the Sybase Shared folders) :

```
PBTRIMCHARCOLUMNS='YES'
```

in the section for your DBMS. With that setting, Pb will trim blanks in datawindows only.

## **GPF when scrolling via the WheelMouse on a datawindow**

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0111.html>

GPF can happen when the Detail band of the datawindow is 0. A workaround is to increase the Detail band at a value higher than 0 (minimum seems to be 4). Also use the latest IntelliMouse driver from [Microsoft](#) which seems to fix the problem.

## **ODBC drivers installed but not visible**

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0115.html>

### **Problem**

Even if an ODBC driver is installed, it's not displayed inside the PB environment so it's not possible to create a DSN using it.

### **Solution**

The PB ODBC drivers list is limited to 1000 characters. Delete unused ODBC drivers from the registry to keep only the good ones.

You can delete entries in the Registry in the branch

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI
```

if a "clean uninstalled" procedure is not available.

## **GPF when debugging with PB7**

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0121.html>

Try to delete the following registry entry :

HKEY\_CURRENT\_USER\Software\Sybase\Powerbuilder\7.0\Layout\Default\Debug

## Avoid Blank sheets when printing reports

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0130.html>

Make sure there are no objects (put there accidentally or by PB) to the left. Stretch the bands to make sure they are not 'hiding' where you can't see them.

## GPF on DW SetFullState (PB6.0)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0136.html>

Using Getfullstate on a server application to fill a DataStore and then passing it to a blob for the client application, it looks like the SetFullstate wasn't working properly or gives some GPF. Specially when the datastore in the client side already have data in it.

A workaround to this problem is to reassign the dataobject property of the datastore to itself before the SetFullState(). Giving something like this :

```
dw_product.dataobject = dw_product.dataobject
```

Thanks to Gildor for the tip!

## Memory usage on SQL Server connection

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0137.html>

The memory used on the client is very large during a connection and on disconnect it comes back down again.

The fix is to put something in the DBParm property of the SQLCA object. Something like "Application='MyApp'" or whatever. In the development environment, edit the connection properties and make sure something is in the Application Name or Workstation Name on the Network tab.

## Accelerator key do not display in W2K

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0193.html>

This is a Windows 2000 setting.

Go into Control Panel -> Display -> Effects  
and turn off the *Hide keyboard navigation indicators until I use the Alt key* setting.

## "Access or violation" error using timestamp with Oracle

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0216.html>

Using the ODBC escape sequence {ts to represent a timestamp value, you may receive the error "Access violation" if your query looks like

```
{ ts '2003-03-25 00:00:00' }
```

With Oracle, you have to add the milliseconds part. So use this instead

```
{ ts '2003-03-25 00:00:00.000000' }
```

## No commit on GPF

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0222.html>

When a transaction disconnects due to severe errors, theit is 'normal' ODBC-Powerbuilder behaviour is to commit the current transaction.

With PB6 (or better), the following DbParm parameter is available. To specify to rollback when you disconnect from the database :

```
SQLCA.DBParm = "CommitOnDisconnect = 'No'"
```

## Hide SqlAnywhere

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0223.html>

When starting the database engine add the "-Q" parameter. Be sure it is in upper case.

## Fix LibraryImport problem

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0244.html>

Thanks to J. Lakeman

When importing a datawindow into your application using LibraryImport, you will sometimes come across a situation where Powerbuilder has a read only file handle on the library you are attempting to import into. In this case the LibraryImport method will fail without returning any error text. This occurs because Powerbuilder caches information and file handles that it has obtained from your libraries. But there is a way to force Powerbuilder to clean them up. Simply open a window.

Powerbuilder caches all of this information while a window is opening as this is the time where multiple class definitions must be loaded and every little bit of speed helps. But when the window has finished opening, these handles will be cleaned up.

```
li_ret = LibraryImport &
  ('library', 'object', ImportDataWindow!, 'syntax', ls_error, 'comments')
IF li_ret<0 AND ls_error=''
THEN
  // a child window that is invisible and will close itself immediately
  open(w_dummy)
  li_ret = LibraryImport &
  ('library', 'object', ImportDataWindow!, 'syntax', ls_error, 'comments')
END IF
```

## Show the underline for keyboard shortcut

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0246.html>

That's standard Windows (XP) behavior.

The underlines won't appear until you press the ALT key.

To disable it and have the underline always visible, right-click the Desktop, choose Properties, and click the Appearance tab. Click the Effects button and remove the check mark from the line *Hide Underlined Letters for Keyboard Navigation Until I Press The Alt Key*.

## Allow user:password in URL

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0249.html>

The following URL syntax is no longer supported in Internet Explorer or in Windows Explorer after you install the MS04-004 Cumulative Security Update for Internet Explorer (832894):

http(s)://username:password@server/resource.ext

This change in the default behavior is also implemented by security updates and service packs that were released after the 832894 security update.

By default, this new default behavior for handling user information in HTTP or HTTPS URLs applies only to Windows Explorer and Internet Explorer. To use this new behavior in other programs that host the Web browser control, create a DWORD value named SampleApp.exe, where SampleApp.exe is the name of the executable file that runs the program. Set the DWORD value's value data to 1 in one of the following registry keys.

- For all users of the program, set the value in the following registry key:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Internet Explorer\  
Main\FeatureControl\FEATURE\_HTTP\_USERNAME\_PASSWORD\_DISABLE

- For the current user of the program only, set the value in the following registry key:

HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\  
Main\FeatureControl\FEATURE\_HTTP\_USERNAME\_PASSWORD\_DISABLE

To disable the new default behavior in Windows Explorer and Internet Explorer, create iexplore.exe and explorer.exe DWORD values in one of the following registry keys and set their value data to 0.

- For all users of the program, set the value in the following registry key:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Internet Explorer\  
Main\FeatureControl\FEATURE\_HTTP\_USERNAME\_PASSWORD\_DISABLE

- For the current user of the program only, set the value in the following registry key:

HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\  
Main\FeatureControl\FEATURE\_HTTP\_USERNAME\_PASSWORD\_DISABLE

ref [Microsoft Article ID : 834489](#)

## Initialization file is not writable. Cannot continue.

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0260.html>

On XP, you may receive a Message Box with the message "*Initialization file is not writable. Cannot continue.*" when starting Powerbuilder in a USER account.

PB is installed in Program Files\Sybase and by default will use the PB.INI located in *Program Files\Sybase\Powerbuilder n.nn* which is a protected area when running with a non-admin account. So PB

can't update its PB.INI file.

You can start PB from an ADMIN account but a better solution is to tell to PB to use an alternate location for PB.INI where regular USER account can write.

Add this key (with regedit)

```
[HKEY_CURRENT_USER\Software\Sybase\PowerBuilder\9.0]
InitPath="the path to user PB.INI eg. c:\dev\pb9"
```

---

---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>

## POWERBUILDER HOW-TO

# Database

---

## pb-database

---

### Suppress ODBC logon window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0006.html>

Use this ConnectOption in your DbParm :

```
SQLCA.DBParm="ConnectionString='DSN=mydb;UID=dba;PWD=sql'," + &
               "ConnectOption='SQL_DRIVER_CONNECT,SQL_DRIVER_NOPROMPT'"
```

### List available ODBC datasources

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0044.html>

You need to declare the following external functions :

```
FUNCTION integer SQLAllocEnv(ref long henv) LIBRARY "odbc32.dll"
FUNCTION integer SQLFreeEnv(long henv) LIBRARY "odbc32.dll"
FUNCTION integer SQLDataSources &
               (long henv, integer idirection, ref string szdsn, int idsnmax, &
                ref integer idsn, ref string szdesc, integer idescmax, ref integer idesc) &
                library "odbc32.dll"
```

The following snippet will initialize a DropDownListbox with DataSources defined on the current workstation.

```
long li_henv
string ls_dsn, ls_desc
integer li_direction, li_dsnmax, li_dsnlen, li_descmax, li_desclen, li_rc
integer li_length = 255

ls_dsn = Space(li_length)
li_dsnmax = li_length
ls_desc = Space(li_length)
li_descmax = li_length
```

```

IF SQLAllocEnv(ll_henv) = -1 THEN
    MessageBox("SQLAllocEnv", "FAILURE")
ELSE
    li_direction = 1
    DO WHILE SQLDataSources &

        (ll_henv, li_direction, ls_dsn, li_dsnmax, li_dsnlen, &
        ls_desc, li_descmax, li_descrlen) = 0
        dd़lb_1.AddItem(ls_dsn + " [" + ls_desc + "]")
    LOOP
    SQLFreeEnv(ll_henv)
END IF

```

## How to get the current DBMS, Database or user through ODBC

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0061.html>

These useful informations can be retrieved via direct calls to the ODBC API. This way we don't need DBMS-specific SELECT statement.

```

[external function declaration]
FUNCTION integer SQLGetInfo  &

    (long hconn, integer infotype, ref string infotypeptr, &
    integer bufferlength, ref integer bufferlengthptr) &

LIBRARY "odbc32.dll"

[powerscript]
string ls_dbms, ls_database, ls_user
integer li_length
CONSTANT integer SQL_DBMS_NAME = 17
CONSTANT integer SQL_DATABASE_NAME = 16
CONSTANT integer SQL_USER_NAME = 47
long ll_dbhandle

ls_dbms = space(256)
ls_database = space(256)
ls_user = space(256)
ll_dbhandle = SQLCA.DbHandle()
SQLGetInfo(ll_dbhandle, SQL_DBMS_NAME, ls_dbms, 255, li_length)
SQLGetInfo(ll_dbhandle, SQL_DATABASE_NAME, ls_database, 255, li_length)
SQLGetInfo(ll_dbhandle, SQL_USER_NAME, ls_user, 255, li_length)

MessageBox("Current DBMS" , trim(ls_dbms))
MessageBox("Current DATABASE" , trim(ls_database))
MessageBox("Current USER" , trim(ls_user))

```

## Create a table from PowerScript

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0024.html>

Use EXECUTE IMMEDIATE. Set Autocommit to true because DDL SQL has to be executed outside of transaction.

```
SQLCA.AutoCommit = True  
ls_sql = "create table #tmp (abc varchar(255))"  
  
EXECUTE IMMEDIATE :LS_SQL USING SQLCA;
```

To alter a table, use the same idea:

```
ls_sql = 'ALTER TABLE dba.tbl_name ADD col_name'  
EXECUTE IMMEDIATE :LS_SQL USING SQLCA;
```

## Prevent creation of PB special table

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0126.html>

These tables (PBCAT\*) are useful only in development, they are not used during run-time. PB will create them automatically if not present in a database. To prevent their creation, add (or modify) the following line in the PB.INI file

```
[Database]  
NoCatalog=1
```

Or using the IDE, go into the DB-Painter, Design, Options and deselect the 'Use extended attributes'.

Thanks to B. Bailey for the following update :

*Actually, PB does use the catalog tables at runtime, if they exist. I have frequently included an ad-hoc query window in many of my applications. As a part of the query window, I create the DW dynamically, using SyntaxFromSQL(). When a DW is thus created, PB uses the catalog tables to define the column characteristics in exactly the same manner as it does in the IDE. For example, if I define an edit style for a column, the dynamic DW will contain the edit style that is specified in the PBCATEDT table.*

## Get time information from the DBMS through ODBC

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0145.html>

This technique is compatible with any ODBC driver.

```
DECLARE dbinfo DYNAMIC CURSOR FOR SQLSA;
ls_sql = "SELECT {fn curdate() }"
// ls_sql = "select {fn curtime() }"
// ls_sql = "SELECT {fn now() }"
PREPARE SQLSA FROM :ls_sql USING SQLCA;

OPEN DYNAMIC dbinfo;
IF SQLCA.SQLCode > 0 THEN
    // erro handling
END IF
FETCH dbinfo INTO :ls_date;
CLOSE dbinfo;
RETURN Date(left(ls_date, 10))
```

## Enable database tracing

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0187.html>

In the pb.ini file set the parameter

```
[database]
dbtracefile=c:\sqltrace.log
```

if not specify, the default is *C:\WINDOWS\PBTRACE.LOG*

You enable the tracing with

```
SQLCA.dbms = "TRACE ODBC" // ODBC
SQLCA.dbms = "TRACE O73" // Oracle 7.3
```

---

---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>

# POWERBUILDER HOW-TO

# Datawindow

---

## pb-datawindow

---

### Load an array with the names of every column in a dw

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0068.html>

```
int colNum, numCols
string colName[]
numCols = Integer(dw_control.Describe("Datawindow.Column.Count"))
FOR colNum = 1 TO numCols
    // Get Column Name with describe
    colName[colNum] = dw_control.Describe("#" + String(colNum) + ".name")
NEXT
```

### Have a different color for newly inserted row

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0069.html>

In the expression painter, code the following for the Background expression :

```
IF ( IsRowNew() , 1090519039, Long(Describe("datawindow.color")) )
```

where 1090519039 is the regular window color.

Using the same idea, to make existing data read-only and newly inserted editable, code the following in the Protect expression :

```
IF ( IsRowNew() , 0 , 1 )
```

### Set a Date to NULL

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0070.html>

```
date ld_date
```

```
SetNull(ld_date)
dw_1.object.datecolumn[row]=ld_date
```

## Create dynamically a DataWindow

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0071.html>

```
string ls_select
string ls_where
string ls_dwsyntax
string ls_err

ls_select = &

"Select id, fname, lname, address, city, state, zip from customer"
ls_where = " where customer.fname like '" + is_cust + "%'"
ls_dwsyntax = SQLCA.SyntaxFromSQL ( ls_select, "Style(Type=grid)", ls_err )
dw_1.Create ( ls_dwsyntax, ls_err )
IF ls_err '' THEN
    MessageBox ( "error - Syntax", ls_err )
ELSE
    dw_1.SetTransObject ( SQLCA )
    dw_1.Retrieve()
END IF
```

## Get the page count of a report

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0072.html>

Put the datawindow into print preview mode

```
dw_1.modify &
( 'datawindow.print.preview=yes' )
```

Get the result of the 'PageCount()' expression

```
ll_PageCount = &
long ( dw_1.describe &
("evaluate('pagecount()', " + string ( dw_1.rowcount() ) + ")"))
```

## Make the ENTER key act as TAB key

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0073.html>

First, define a user event to correspond with the *pbm\_dwnprocessenter* event on a datawindow. Then in that event :

```
Send(Handle(this), 256, 9, Long(0,0))
RETURN 1
```

## Display only distinct rows

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0074.html>

First you need to Sort() the datawindow based on the columns that need to be distinct. After you apply a Filter(), to filter out the non-distinct rows.

For example, we have a datawindow with a column named *prod\_id* and we want to display only one row for each *prod\_id*.

First we turn off the datawindow redraw function to speed up the operation

```
dw_1.setredraw(false)
```

Sort the rows based on the *prod\_id* column

```
dw_1.setsort("prod_id a")
dw_1.sort()
```

Define the appropriate Filter

```
dw_1.SetFilter("IsNull(prod_id[-1]) OR prod_id[-1] = prod_id")
dw_1.filter()
```

Finally turn back on the datawindow redraw

```
dw_1.setredraw(true)
```

## Extract the display value of DDDW

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0075.html>

```
string ls_dept
long ll_row

ll_row = 3 // we want the displayed value of dept_id in row 3

ls_dept = dw_1.Describe &

("Evaluate('LookupDisplay(dept_id)', " + String(ll_row) + ")")

// LS_DEPT == DEPARTMENT NAME FROM THE DDDW
```

## Do simple validation with the Datawindow properties

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0076.html>

digit only, length unknown

```
[validation tab]
match(gettext(), '^[0-9]+$')
```

digit only, length fixed

```
[edit tab]
style : EditMask mask : #####
```

```
[validation tab]
len(gettext())=5
```

digit, '%' or "\_"

```
[validation tab]
match(gettext(), '^[0-9%_]+$')
```

digit only and must end with "%"

```
[validation tab]
match(gettext(), '^[0-9]+[%]$')
```

canadian postal code

```
[edit tab]
style : EditMask mask : !#! #!#
```

```
[validation tab]
match(gettext(), '^[a-z][0-9][a-z][0-9][a-z][0-9]$')
```

## Set a default value for a DDDW

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0077.html>

```
dw_1.Object.[colname].Initial = ""
```

## Delete current selection in DDDW

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0078.html>

Create a user event in the datawindow control mapped to pbm\_dwnkey. Put the following code in the event:

```
string ls_columnname
string ls_null
long ll_null
long ll_row

// for a dddw, the delete key deletes the current selection(replace by NULL)
if key = KeyDelete! then
  ll_row = this.GetRow()
  if ll_row > 0 then
    ls_columnname = this.GetColumnName()
    if not (IsNull(ls_columnname) or ls_columnname = '') then
      if this.Describe(ls_columnname + ".Edit.Style") = 'dddw' and + &
        this.describe(ls_columnname + ".DDDWN.IIsNotNull") = 'yes' then
          if left(this.Describe(ls_columnname + ".coltype"), 4) = 'char' then
            SetNull(ls_null)
            this.SetItem(ll_row,ls_columnname,ls_null)
          else
            SetNull(ll_null)
            this.SetItem(ll_row,ls_columnname,ll_null)
          end if
        end if
      end if
    end if
  end if
end if

return 0
```

## Alternate row color

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0079.html>

Place a rectangle with a transparent background color. Place fields on the rectangle. In the expression tab for the rectangle, in the background color field :

```
if ( mod(getrow(),2) = 0, oneColor, anotherColor )
```

## Pass the content of a dw to another window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0080.html>

Call the second window passing a reference of the datawindow

```
OpenWithParm( w_window, dw_1 )
```

then in the Open event of w\_window

```
datawindow ldw_parm  
ldw_parm = Message.PowerObjectParm  
dw_1.DataObject = ldw_parm.DataObject  
ldw_parm.ShareData(dw_1)
```

and in the Close event

```
dw_1.ShareDataOff()
```

## Detect which button is clicked in a dw (PB6)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0081.html>

```
[ButtonClicked event]  
string button_name  
button_name = dwo.name  
CHOOSE CASE button_name  
CASE "cb_ok"  
    MessageBox("Testing", "cb_test is clicked")  
CASE "cb_cancel"
```

```
    MessageBox("Testing", "cb_test2 is clicked")
END CHOOSE
```

## Detect if a row is new

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0082.html>

```
dwitemstatus      rowstatus
IF dw.GetRow() > 0 THEN
    rowstatus = dw.GetItemStatus( dw.GetRow(), 0, Primary! )
    IF rowstatus = New! OR rowstatus = NewModified! THEN
        // new row
    ELSE
        // not new
    END IF
END IF
```

## Set cursor at end of text in a dwcolumn

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0083.html>

```
dw_1.SelectText (len (column_name) + 1, 0)
```

## Make a static text visible on the last page only

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0084.html>

In visible property on the Expression Tab

```
IF (Page() = PageCount(), 1, 0)
```

## Print to a file

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0085.html>

```
dw_1.Object.DataWindow.Print.Filename = 'report.prn'  
dw_1.Print()
```

## Make a dw with a title unmoveable

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0086.html>

```
[other event]  
uint a,b  
  
a = Message.WordParm  
CHOOSE CASE a  
CASE 61458 // 61456 maybe necessary...  
    Message.Processed = true  
    Message.ReturnValue = 0  
END CHOOSE
```

An other way is to map *pbm\_nclbuttondown* event on your datawindow, and code the mapped event :

```
uint HTCAPTION = 2  
IF hittestcode = HTCAPTION THEN  
    Message.processed = true  
    RETURN 1  
END IF
```

## Trap the click on down Arrow on DDLB

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0087.html>

```
[ue_opendddw mapped to pbm_dwndropdown]  
messagebox("DropDownOpened",this.getColumnName())
```

## Scroll 2 datawindows in sync

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0088.html>

```
[ScrollVertical event of dw_1]  
dw_2.Object.datawindow.verticalscrollposition = scrollpos
```

## Get the current filter

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0089.html>

```
ls_filter = dw_1.describe("datawindow.table.filter")
IF ls_filter = "?" THEN
    MessageBox("Oups!", "no filter defined")
END IF
```

## Disable a dw button

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0095.html>

```
ls_result = dw_1.Modify("cb_1.SuppressEventProcessing=Yes")
or
dw_1.object.cb_1.SuppressEventProcessing ='Yes'
```

## Prevent vertical scrolling via the mouse wheel or keyboard

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0090.html>

For the wheel, declare an user event, *ue\_noscroll*, mapped to *pbm\_vscroll*.

```
[ue_noscroll]
// prevent vertical scrolling
return 1
```

For the keyboard, map the *pmb\_dwnkey* event to user event *ue\_dwnkey*.

```
[ue_dwnkey]
CHOOSE CASE key
CASE KeyEnter!, KeyPageUp!, KeyPageDown!, KeyUpArrow!, KeyDownArrow!
    This.AcceptText()
    RETURN 1
END CHOOSE
RETURN 0
```

## Cut and Paste datawindow content

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0091.html>

Datawindow's SaveAs function can save the data directly to the clipboard.

```
dw_1.saveas ( "data.txt", clipboard!, true)
```

Then simply open another program like NotePad or Excel and paste the data.

See this [related howto](#)

See also this [related howto](#)

## Open a DDDW via Powerscript

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0092.html>

```
[external function declaration]
SUBROUTINE keybd_event( int bVk, int bScan, int dwFlags, int dwExtraInfo) &
LIBRARY "user32.dll"

[powerscript]
constant integer VK_F4 = 115

dw_1.SetFocus()
dw_1.SetColumn( "dept_head_id" ) // the DDDW
keybd_event( VK_F4,0,0,0 ) // F4 key down
keybd_event( VK_F4,0,2,0 ) // F4 key up
```

## Change the presentation style

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0093.html>

1. From the Library painter, export the datawindow to a file.
2. Edit the file and locate the "PROCESSING" parameter. Possible values are :
  - ◆ 0 (Default) Form, group, query, or tabular
  - ◆ 1 Grid
  - ◆ 2 Label

- ◆ 3 Graph
- ◆ 4 Crosstab
- ◆ 5 Composite
- ◆ 7 RichText

3. To change from a tabular to grid :

```
processing=0 CHANGE TO processing=1
```

4. Save the change and import back in a library.

## Autoselect an EditMask

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0112.html>

```
[ItemFocusChanged event]
CHOOSE CASE this.Describe(dwo.name + '.editmask.mask')
CASE '?' , '!'
CASE ELSE
    this.SelectText(1, 999)
END CHOOSE
END IF

RETURN 0
```

## Sort an array

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0114.html>

We create a small datastore and use it to sort our array.

```
datastore lds_temp
string ls_err
integer i

// change the datastore definition according to the array data type
long ll_array[] = { 2 , 3, 6, 5 }
string ls_dsdef = &

'release 6; datawindow() table(column=(type=long name=a dbname="a") )'

lds_temp = CREATE datastore
lds_temp.Create(ls_dsdef, ls_err)
// put the array in the datastore
lds_temp.object.a.current = ll_array
lds_temp.SetSort ("a ASC")
lds_temp.Sort()
```

```
// get back the array
ll_array = lds_temp.object.a.current

FOR i = 1 to Upperbound(ll_array)
    MessageBox("", string(ll_array[i]))
NEXT

DESTROY lds_temp
```

## Get the count of rows selected

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0183.html>

```
long ll_Selected

ll_Selected = &

long(dw_1.describe("evaluate('sum( if(isselected(), 1, 0) for all)',1)"))
```

## Get data from Excel via the clipboard

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0200.html>

```
OLEObject excel

Integer li_RetValue, li_rtn
Boolean lb_sheet_rtn
Long ll_cnt

excel = create OLEObject

li_rtn = excel.ConnectToNewObject("excel.application")
IF li_rtn 0 THEN
    MessageBox('Excel erro', 'can not run Excel Program')
    DESTROY excel
    RETURN 0
END IF

excel.WorkBooks.Open( "c:\mysheet.xls" )
excel.Application.Visible = false
excel.Windowstate = 2 // 1 : Normal, 2 : Minimize, 3 : Maximize
```

```
lb_sheet_rtn = excel.worksheets(1).Activate
excel.Worksheets(1).Range("A1:E5000").Copy // copy to clipboard
ll_cnt = dw_1.importclipboard()
IF ll_cnt <= 1 THEN
    Messagebox("Inf", "Could not find .")
END IF

excel.Worksheets(1).Range("A10000:A10000").Copy //reset clipboard
excel.Application.Quit
excel.DisConnectObject()
DESTROY excel
```

See this [related howto](#)

See also this [related howto](#)

## Make a datawindow static text flash

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0217.html>

On the General tab of the datawindow object, you set the Timer Interval value (say 2 for 2 seconds)and for a static text, you add this expression in the Visible property :

```
If( Mod( Second( Now() ), 2 ) = 0, 1, 0 )
```

## Suppress datawindow built-in popup

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0218.html>

Powerbuilder pops up its own MessageBox when an error is detected in an Evaluate() or Filter() expression. If you want to suppress this kind of interaction and deal with the error yourself, do

```
dw_1.Modify ("DataWindow.NoUserPrompt='yes '")
```

## Print a datawindow into a PDF

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0235.html>

PB9 offers some support to print a datawindow into a PDF. This is possible with the help of Ghostscript.

Ghostscript version 8.14 allows free use, copying, and distribution by end users, but does not allow commercial distribution. More recent version have a GPL type license which maybe not appropriate, you need to read the license!

1. First download Ghostscript 8.14 (gs814w32.exe) from <http://ghostscript.com/doc/AFPL/index.htm>.
2. Read the end use license agreement
3. Install Ghostscript and make sure that the PATH contains the Ghostscript bin directory.
4. Before printing a datawindow into a PDF, you must
  1. Open the dataobject.
  2. In the properties window go to the data export tab.
  3. Select PDF from the "Format to configure" drop down.
  4. Make sure the "Method" drop down has Distill! in it.

Then to print with a the dialog and save:

```
dw_1.SaveAs()
```

And to save without a dialog:

```
dw_1.SaveAs('c:\file.pdf', PDF!, FALSE)
```

## View datawindow content while in DEBUG mode

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/powerbuilder-see-datawindow-in-debug-mode.html>

It's very strange that Powerbuilder does not provide an easy way to examine a datawindow or datastore current data when running in the debugger.

One way is to add a **watch** variable on the datawindow/datastore pointer with this expression.

```
[dw or ds instance variable].object.datawindow.data
```

to show in the watch window the datawindow/datastore content. But the output is very "raw"...

A better way is to use to same idea but save the current datawindow/datastore data to a file.

```
[dw or ds instance variable].saveas("c:\mydatawindow.dat", text!, true)
```

At each step, the file will be updated with the current data.

```
fn_item_code fn_item_type_code hilmt_super_flag
ADMIN_ACCESS_D_ADDRESS MODE N
ADMIN_ACCESS_D_ADDRESS MODE N
ADMIN_ACCESS_D_ADDRESS MODE N
ADMIN_ACCESS_D_BROKER MODE N
ADMIN_ACCESS_D_BROKER MODE N
```

```
ADMIN_ACCESS_D_BROKER      MODE      N  
...
```

You can use like BareTail to see the update live!

---

---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>

## **POWERBUILDER HOW-TO**

# PFC

---

## pb-pfc

---

### Disconnect an inactive application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0097.html>

```
[application idle event]
gnv_app.event pfc_idle()

[appmanager contructor event]
Idle(3600) // that's 3600 seconds

[appmanager pfc_idle event]
IF SQLCA.of_IsConnected() THEN
    SQLCA.of_Disconnect()
    // HALT CLOSE the application
    // or call the pfc_logon event
END IF
```

### Display compile timestamp in the w\_about window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0098.html>

```
[appmanager contructor event]
constant string ls_version =  &

'Application version 1.0~r~nCompile Date: ' + &
string(Today(), 'mmm dd, yyyy hh:mm')
THIS.OF_SETVERSION(LS_VERSION)
```

Another way way is to look at the executable file and extract the last modified date attribute.

```
date dt
time tm

f_setfilesrv(lnv_fs, True)
lnv_fs.of_GetLastWriteDateTime &
```

```
(gnv_app.iapp_object.appname + ".exe", dt, tm)
st_build.text = &

"Built on "+string(dt, "mm/dd/yyyy")+" at "+string(tm,"hh:mmam/pm")
DESTROY lnv_fs
```

## Use the w\_master pfc\_Save event

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0099.html>

The w\_master pfc\_Save event will update all "self\_updateable" objects. u\_dw is self updateable by default but you must of\_SetUpdateable(TRUE) n\_ds, u\_lvs, u\_tab and u\_tcs objects to enable this feature. Each these self-updatable objects have their own events that will be called by the w\_master pfc\_Save event. Here is the event sequence that will occur during a pfc\_Save

<b>pfc_AcceptText</b>	execute an AcceptText on all self-updating objects
<b>pfc_UpdatesPending</b>	Check if there is any self-updating object that need to be updated
<b>pfc_Validation</b>	extend on the object  performs validation on the object level returns SUCCESS if ok, FAILURE otherwise. For non-PFC object, create an user event called ue_validation which returns an integer greater or equals to 0 for SUCCESS.
<b>pfc_UpdatePrep</b>	extend on the window  <i>Empty event</i> , use to prepare an update
<b>pfc_PreUpdate</b>	extend on the window or control  <i>Empty event</i> more validations if necessary, return SUCCESS to continue the pfc_Save process or FAILURE to abort
<b>pfc_BeginTran</b>	extend on the window  <i>Empty event</i> code the transaction beginning, returns SUCCESS to continue the update
<b>pfc_Update</b>	extend on the window

	Do the update on the self-updating objects. For non-self-updating object, you can code here the update procedure. Returns SUCCESS or FAILURE. If it's a failure, use of_SetDBErrorMsg() to set an appropriate error message (it will be displayed by the pfc_DBError error).
<b>pfc_EndTrand</b>	extend on the window  <i>Empty event</i> Here you can code the SQLCA.of_Commit() or SQLCA.of_Rollback() depending on the savecode (an event parameter) value.
<b>pfc_DBError</b>	extend or override on the window  If an error occurs, this event will occur.
<b>pfc_PostUpdate</b>	extend on the window or control  Reset update flags on the self-updateable objects

## Control the Cut&Paste items in m\_master

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0100.html>

[Edit menu item from w\_master, Clicked event]

```

graphicobject  lgo_current_object
boolean        lb_text_selected = false
boolean        lb_can_modify_control = false
datawindow     ldw_current
editmask       lem_current
singlelineEdit lsle_current
multilineedit  lmle_current
dropdownlistbox lddlb_current
string         ls_columnname, ls_temp

m_master.m_edit.m_undo.enabled = false
m_master.m_edit.m_copy.enabled = false
m_master.m_edit.m_cut.enabled = false
m_master.m_edit.m_paste.enabled = false
m_master.m_edit.m_clear.enabled = false

lgo_current_object = GetFocus()
IF IsNull( lgo_current_object ) THEN RETURN

CHOOSE CASE lgo_current_object.TypeOf()
CASE multilineedit!
  lmle_current = lgo_current_object
  m_master.m_edit.m_undo.enabled = lmle_current.CanUndo()
  lb_text_selected = lmle_current.SelectedLength() > 0

```

```

lb_can_modify_control = NOT lmle_current.displayonly
CASE editmask!
  lem_current = lgo_current_object
  m_master.m_edit.m_undo.enabled = lem_current.CanUndo()
  lb_text_selected = lem_current.SelectedLength() > 0
  lb_can_modify_control = true
CASE singlelineedit!
  lsle_current = lgo_current_object
  m_master.m_edit.m_undo.enabled = lsle_current.CanUndo()
  lb_text_selected = lsle_current.SelectedLength() > 0
  lb_can_modify_control = NOT lsle_current.displayonly
CASE DataWindow!
  ldw_current = lgo_current_object
  m_master.m_edit.m_undo.enabled = ldw_current.CanUndo()
  lb_text_selected = ldw_current.SelectedLength() > 0
  ls_columnname = ldw_current.GetColumnName ()
  IF ls_columnname "" THEN
    ls_temp = ldw_current.Describe( ls_columnname + &
      ".edit.style")
    CHOOSE CASE ls_temp
    CASE "edit"
      ls_temp = ldw_current.Describe( ls_columnname + &
        ".edit.displayonly")
      lb_can_modify_control = (ls_temp = "no")
    CASE "ddlb", "dddw"
      ls_temp = ldw_current.Describe( ls_columnname + &
        ".." + ls_temp + ".allowedit")
      lb_can_modify_control = ls_temp = "yes"
    CASE "editmask"
      lb_can_modify_control = true
    END CHOOSE
  END IF
CASE DropDownListBox!
  lddb_current = lgo_current_object
  lb_text_selected = lddb_current.SelectedLength() > 0
  lb_can_modify_control = lddb_current.allowedit
END CHOOSE

IF lb_text_selected = true THEN
  m_master.m_edit.m_copy.enabled = true
  m_master.m_edit.m_cut.enabled = lb_can_modify_control
  m_master.m_edit.m_clear.enabled = lb_can_modify_control
ELSE
  m_master.m_edit.m_cut.enabled = false
  m_master.m_edit.m_copy.enabled = false
  m_master.m_edit.m_clear.enabled = false
END IF

m_master.m_edit.m_paste.enabled = Len( ClipBoard() ) > 0 AND &
  lb_can_modify_control

```

## Have PFC Help accessible through the Toolbar

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0101.html>

Mouse right click on the top toolbar, choose customize. In the "Select palette" section, select the "customize" radio button. Then drag an icon to the current toolbar. In the command line field (adjust the path for your current PB installation), type :

```
WINHELP C:\PROGRAM FILES\SYBASE\PB6\HELP\PBPFC60.HLP
```

In the Item Text and Item microhelp, type :

```
PFC HELP
```

NOTE: The WinHelp utility has been renamed to WinHlp32 in W2K.

## Have a FileExists function with wildcard

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0102.html>

```
[boolean of_FileExists(string as_file)]  
  
Integer li_entries  
n_cst_filesrwin32 lnv_fsrv  
n_cst_dirattrib lnv_dirlist[]  
  
lnv_fsrv = CREATE n_cst_filesrwin32  
li_entries = lnv_fsrv.of_DirList(as_file, 0, lnv_dirlist)  
DESTROY lnv_fsrv  
  
RETURN (li_entries > 0 )
```

## Specify objects to be save during a pfc\_Save event

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0103.html>

The pfc\_Save event uses the Window control array (by default) to get what are the objects to be updated and in what order. You can use the of\_SetUpdateObjects() function to specify your array of objects to be updated. For example, the following will make sure that dw\_2 is updated before dw\_1.

```
this.of_SetUpdateObjects({dw_2, dw_1})
```

The same function is used to include objects which are not self-updateable by default, like n\_ds for example.  
Be sure to call of\_SetUpdateable(true) in the n\_ds constructor.

```
this.of_SetUpdateObjects({dw_1, dw_2, ids_data})
```

## Detect PFC version

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0104.html>

The *n\_cst\_debug* contains CONSTANT definitions to show different informations about the current PFC installation.

```
PFC_BUILD_DATE      Build date Date
PFC_FIXES          PFC revision level Integer
PFC_MAJOR          PFC major revision level Integer
PFC_MINOR          PFC minor revision Integer
PFC_NAME           PFC title String
PFC_BUILD_TIME     Build time Time
```

Since they are CONSTANTS, you don't need to instantiate an object simply to get these values, just refer to them with *n\_cst\_debug.[CONSTANT NAME]*. This can be useful to show in the w\_about window for example..

```
string ls_ver = string(n_cst_debug.PFC_MAJOR) + "." + &
               string(n_cst_debug.PFC_MINOR) + "." + &
               string(n_cst_debug.PFC_FIXES)
MessageBox(n_cst_debug.PFC_NAME, ls_ver)
```

---

---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>

# POWERBUILDER HOW-TO

# Powerscript

---

## pb-powerscript

---

### Get the PBL name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0201.html>

Get the name of current PBL (plus its full path)

```
ClassDefinition lcd  
  
lcd = this.ClassDefinition  
MessageBox("", "My PBL name is " + lcd.LibraryName )
```

---

See also this [PB HowTo](#)

### Time the execution

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0010.html>

Use the CPU() function for timing.

```
long ll_start, ll_elapsed  
ll_Start = CPU ( )  
/*  
**your code to time test  
*/  
ll_elapsed = CPU ( ) - ll_start
```

### Rename an object

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0011.html>

Simply open the object to be renamed via the Library painter and do a Save As... with the new name. Change any existing references to the original name to the new name in any other objects and regenerate. Delete the original object.

## Get the numeric value for a color

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0018.html>

(from the pfc\_n\_cst\_color class)

```
BUTTONFACE = 78682240
WINDOW_BACKGROUND = 1087434968
WINDOW_TEXT = 33554592
APPLICATION_WORKSPACE = 268435456
TRANSPARENT = 553648127

BLACK = RGB(0, 0, 0)
WHITE = RGB(255, 255, 255)
LIGHT_GRAY = RGB(192, 192, 192)
DARK_GRAY = RGB(128, 128, 128)
RED = RGB(255, 0, 0)
DARK_RED = RGB(128, 0, 0)
GREEN = RGB(0, 255, 0)
DARK_GREEN = RGB(0, 128, 0)
BLUE = RGB(0, 0, 255)
DARK_BLUE = RGB(0, 0, 128)
MAGENTA = RGB(255, 0, 255)
DARK_MAGENTA = RGB(128, 0, 128)
CYAN = RGB(0, 255, 255)
DARK_CYAN = RGB(0, 128, 128)
YELLOW = RGB(255, 255, 0)
BROWN = RGB(128, 128, 0)
```

These values are hard-coded, if you need more flexibility, you can query Windows to dynamically get the current color setting.

```
[External function declaration]
FUNCTION unsignedlong GetSysColor(int nIndex) LIBRARY "user32.dll"
```

using the following parameters :

COLOR_SCROLLBAR	0
COLOR_BACKGROUND	1
COLOR_ACTIVECAPTION	2
COLOR_INACTIVECAPTION	3
COLOR_MENU	4
COLOR_WINDOW	5
COLOR_WINDOWFRAME	6
COLOR_MENUTEXT	7
COLOR_WINDOWTEXT	8
COLOR_CAPTIONTEXT	9
COLOR_ACTIVEBORDER	10
COLOR_INACTIVEBORDER	11

COLOR_APPWORKSPACE	12
COLOR_HIGHLIGHT	13
CLOR_HIGHLIGHTTEXT	14
COLOR_BTNFACE	15
COLOR_BTNSHADOW	16
COLOR_GRAYTEXT	17
COLOR_BTNTTEXT	18
COLOR_INACTIVECAPTIONTEXT	19
COLOR_BTNHIGHLIGHT	20
// (WINVER >= 0x0400)	
COLOR_3DDKSHADOW	21
COLOR_3DLIGHT	22
COLOR_INFOTEXT	23
COLOR_INFOBK	24
COLOR_DESKTOP	COLOR_BACKGROUND
COLOR_3DFACE	COLOR_BTNFACE
COLOR_3DSHADOW	COLOR_BTNSHADOW
CLOR_3DHIGHLIGHT	COLOR_BTNHIGHLIGHT
COLOR_3DHIGHLIGHT	COLOR_BTNHIGHLIGHT
COLOR_BTNHIGHLIGHT	COLOR_BTNHIGHLIGHT

## Retrieve an environment variable

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0020.html>

```
// PB6
ContextKeyword lcxk_base
string ls_Path
string ls_values[]

this.GetContextService("Keyword", lcxk_base)
lcxk_base.GetContextKeywords("path", ls_values)
IF Upperbound(ls_values) > 0 THEN
    ls_Path = ls_values[1]
ELSE
    ls_Path = "*UNDEFINED*"
END IF
```

See this [Howto for common XP environment variables](#)

## Common XP environment variables

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0254.html>

ALLUSERSPROFILE	location of the All Users Profile.
APPDATA	location where applications store data by default.
CD	current directory string.
CLIENTNAME	client's NETBIOS name when connected to terminal server session.
CMDCMDLINE	command line used to start the current cmd.exe.
CMDEXTVERSION	version number of the current Command Processor Extensions.
CommonProgramFiles	path to the Common Files folder.
COMPUTERNAME	name of the computer.
COMSPEC	path to the command shell executable.
DATE	current date.
ERRORLEVEL	error code of the most recently used command.
HOMEDRIVE	drive letter is connected to the user's home directory.
HOMEPATH	full path of the user's home directory.
HOMESHARE	network path to the user's shared home directory.
LOGONSEVER	name of the domain controller that validated the current logon session.
NUMBER_OF_PROCESSORS	number of processors installed on the computer.
OS	name of the operating system. (Windows XP and Windows 2000 list the operating system as Windows_NT.)
Path	search path for executable files.
PATHEXT	file extensions that the operating system considers to be executable.
PROCESSOR_ARCHITECTURE	processor's chip architecture.
PROCESSOR_IDENTIFIER	description of the processor.
PROCESSOR_LEVEL	model number of the computer's processor.
PROCESSOR_REVISION	revision number of the processor.
ProgramFiles	path to the Program Files folder.
PROMPT	command-prompt settings for the current interpreter.
RANDOM	random decimal number between 0 and 32767.
SESSIONNAME	connection and session names when connected to terminal server session.
SYSTEMDRIVE	drive containing the Windows root directory.
SYSTEMROOT	location of the Windows root directory.
TEMP and TMP	default temporary directories for applications that are available to users who are currently logged on.
TIME	current time.
USERDOMAIN	name of the domain that contains the user's account.
USERNAME	name of the user currently logged on.
USERPROFILE	location of the profile for the current user.
WINDIR	location of the OS directory.

To read an environment variable, see these HowTo's for [Powerbuilder](#), [Java](#) and [VBScript](#).

## Blank an array

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0022.html>

Simply set it equal to an array of the same type.

```
integer temp[30]
integer blank[30]

temp[1] = 100
```

```
temp = blank
```

## Make a reference to an instance variable from a Menu

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0027.html>

You want to access a variable called, ii\_id, defined in the window w\_MyWindow from the menu attached to it.

```
w_MyWindow = ParentWindow  
ls_id = w_mywindow.ii_id
```

## Make a reference to a Menu from a frame or a sheet

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0028.html>

Keep in mind that references to a menu should always be based on the window (or frame/sheet) it is attached to, not directly.

```
my_menu m_menu // or w_master if PFC are used  
  
m_menu = w_frame.MenuId  
m_menu.m_file.m_retrieve.toolbaritemvisible = FALSE
```

## Get the Frame from a sheet

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0116.html>

```
w_frame w_parent  
w_parent = mySheet.ParentWindow()
```

## Get the remainder of a decimal

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0033.html>

```
decimal d1 = 5.25
decimal d2
d2 = abs(truncate(d1,0) - d1)
MessageBox("",string(d2))
```

## Detect PB version

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0054.html>

```
string ls_PBver
environment env
GetEnvironment (env)

ls_PBver = string(env.pbmajorrevision) + '.' + &
           string(env.pbminorrevision) + '.' + &
           string(env.pbfixedesrevision)
```

## Make a window stay "on top"

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0025.html>

```
windowname.SetPosition(TopMost!)
```

## Attach a menu in a response window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0036.html>

Use the ChangeMenu function in the open event to attach a menu.

```
[open event]
ChangeMenu(m_master)
```

## Navigate through a Tab control with keyboard

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0038.html>

If the window type is Response or Main, CTRL TAB and CTRL SHIFT TAB will navigate through the Tab pages. With the Sheet type, you have to simulate the navigation by declaring a shortcut in the menu and then use the SelectTab() function.

## Remove the rtf information from the rte content

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0040.html>

To transfer the rte control to mle control, try this :

```
String ls_temp  
rte_1.SelectTextAll()  
ls_temp = rte_1.SelectedText()  
mle_1.Text = ls_temp
```

or use the clipboard

```
rte_1.SelectTextAll()  
rte_1.Copy()  
mle_1.Paste()
```

## Capitalize a name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0045.html>

For example, to change "john smith" to "John Smith", you can use datawindow function WordCap()

```
ls_string ='john smith'  
ls_string = dw_1.Describe ( "Evaluate('WordCap(~'" + ls_string + "~')',0)")
```

Or use the appropriate database function (if supported), for Oracle it looks like

```
SELECT InitCap('john smith') into :ls_name  
FROM dual USING SQLCA;
```

For PFC, use the of\_WordCap() method from the n\_cst\_string object.

## Convert an hex string to its decimal equivalent

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0048.html>

```
[of_hex2long(as_hex) returns a long]
string ls_hex
integer i,length
long result = 0

length = len(as_hex)
ls_hex = Upper(as_hex)
FOR i = 1 to length
    result += &
        (Pos ('123456789ABCDEF', mid(ls_hex, i, 1)) * &
        ( 16 ^ ( length - i ) ))
NEXT
RETURN result
```

See this [HowTo](#) for long2hex conversion.

## Have a window close itself after a delay

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0051.html>

```
[open event]
Timer(20) // 20 seconds

[timer event]
Timer(0)
CLOSE(THIS)
```

## Have a "mouse-over" on a component

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0042.html>

Suppose you have a static text on a Window. When the mouse is over it, the static text foreground color change to red. When the mouse quit the static text, the foreground color is black.

```
[window mousemove event]

// just to display something
st_1.text = string(xpos) + ", " + string(ypos)

IF xpos >= st_1.X AND (xpos <= st_1.x + st_1.Width) AND &
```

```

    ypos >= st_1.y AND (ypos <= st_1.y + st_1.Height) THEN
    st_1.textcolor = 255
ELSE
    st_1.textcolor = 0
END IF

```

## Convert a number to hexadecimal

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0059.html>

```

[of_long2hex(long alnumber, integer ai_digit) returns a string]

long ll_temp0, ll_temp1
char lc_ret

IF ai_digit > 0 THEN
    ll_temp0 = abs(al_number / (16 ^ (ai_digit - 1)))
    ll_temp1 = ll_temp0 * (16 ^ (ai_digit - 1))
    IF ll_temp0 > 9 THEN
        lc_ret = char(ll_temp0 + 55)
    ELSE
        lc_ret = char(ll_temp0 + 48)
    END IF
    RETURN lc_ret +
        of_long2hex(al_number - ll_temp1 , ai_digit - 1)
END IF
RETURN ""

// of_longhex(256, 4) returns "0100"
// of_longhex(256, 3) returns "100"

```

See this [HowTo](#) for hex2long conversion.

## Evaluate an expression

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0060.html>

The datastore can be useful to evaluate a numerical expression at run-time. Simply create a minimal datastore and evaluate the expression.

```
datastore lds_temp
```

```
string ls_result
string ls_expection

string ls_err
// a dummy minimal datastore definition
string ls_dsdef = &

'release 6; datawindow() table(column=(type=char(255) name=a dbname="a") )'

lds_temp = CREATE datastore
lds_temp.Create(ls_dsdef, ls_err)
lds_temp.InsertRow(0)

IF Len(ls_err) > 0 THEN
    MessageBox("ds create", ls_err)
ELSE
    ls_expection = "(2 + 7) / 3"
    ls_result = &

    lds_temp.Describe ('evaluate( " ' + ls_expection + ' " ,1)' )
    MessageBox("ds evaluate", ls_result)
END IF
DESTROY lds_temp
```

## Detect if running in PB or executable

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0067.html>

```
IF Handle(GetApplication()) = 0 THEN
    MessageBox("Info", "Running in PB environment")
ELSE
    MessageBox("Info", "Running in standalone executable")
END IF
```

## Change text on a button to bold

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0113.html>

```
cb_1.weight = 700 // bold
cb_1.weight = 400 // normal
```

## Sort an array

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0114.html>

We create a small datastore and use it to sort our array.

```
datastore lds_temp
string ls_err
integer i

// change the datastore definition according to the array data type
long ll_array[] = { 2 , 3, 6, 5 }
string ls_dsdef = &

    'release 6; datawindow() table(column=(type=long name=a dbname="a") )'

lds_temp = CREATE datastore
lds_temp.Create(ls_dsdef, ls_err)
// put the array in the datastore
lds_temp.object.a.current = ll_array
lds_temp.SetSort("a ASC")
lds_temp.Sort()
// get back the array
ll_array = lds_temp.object.a.current

FOR i = 1 to Upperbound(ll_array)
    MessageBox("", string(ll_array[i]))
NEXT

DESTROY lds_temp
```

## Get data from the Internet

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0128.html>

Create a new userobject from standard class **internetresult**, call it n\_cst\_internet.

Declare this instance variable

```
string is_data
```

With that userobject type, you need to overload the internetdata function (which will be called by the PB VM automatically). The return type is integer and the parameter is data, its type is a blob.

```
[integer internetdata(blob data)]
```

```
is_data = string(data)
RETURN 1
```

NOTE : Since PB10 is unicode-based (previous version were ansi-based), you may need to specify that result contains Ansi characters and not the default Unicode encoding.

```
[integer internetdata(blob data)]
is_data = String(data, EncodingANSI!)
RETURN 1
```

Then from your script

```
integer li_rc
inet linet_main
n_cst_internet luo_data // as defined above

linet_main = CREATE inet
luo_data = CREATE n_cst_internet

SetPointer(HourGlass!)
li_rc = &

linet_main.GetURL("http://www.rgagnon.com/thanks.html", luo_data)
SetPointer(Arrow!)
IF li_rc = 1 THEN
    MessageBox("Data from Real's HowTo", luo_data.is_data)
ELSE
    MessageBox("Data from Real's HowTo", "Oops rc:" + string(li_rc))
END IF

DESTROY luo_data
DESTROY linet_main
```

POSTing data maybe a little more tricky. You may need to set all the headers.

```
string httprequest,ls_header
String ls_url,ls_headers
long ll_ret
long ll_length
Blob lblb_args
inet linet_main
n_cst_internet luo_data

linet_main = CREATE inet

luo_data = CREATE n_cst_internet

ls_url = "http://localhost/mypage.jsp"
lblb_args = blob("foo=bar")
ll_length = Len(lblb_args)
```

```

ls_headers = "Content-Type: " + &
             "application/x-www-form-urlencoded~n" + &
             "Content-Length: " + String( ll_length ) + "~n~n"
ll_ret = libet_main.PostURL(ls_url, lblb_args, ls_headers, 8080, luo_data)

```

Since PB10 is using Unicode to encode characters, code designed with PB9 (which is Ansi) with the PostURL function may not work. The fix is simple, specify the Ansi encoding in the Blob function.

```
lblb_args = Blob("foo=bar", EncodingAnsi!)
```

## Transform a KeyCode! to a String

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0129.html>

```

// returns a String corresponding to the KeyCode! received as
// argument (akc_key)

string ls_ret = ""

CHOOSE CASE akc_key
CASE Key0!, KeyNumPad0!
  ls_ret ="0"
CASE Key1!, KeyNumPad1!
  ls_ret ="1"
CASE Key2!, KeyNumPad2!
  ls_ret ="2"
CASE Key3!, KeyNumPad3!
  ls_ret ="3"
CASE Key4!, KeyNumPad4!
  ls_ret ="4"
CASE Key5!, KeyNumPad5!
  ls_ret ="5"
CASE Key6!, KeyNumPad6!
  ls_ret ="6"
CASE Key7!, KeyNumPad7!
  ls_ret ="7"
CASE Key8!, KeyNumPad8!
  ls_ret ="8"
CASE Key9!, KeyNumPad9!
  ls_ret ="9"
CASE KeySpaceBar!
  ls_ret = " "
END CHOOSE

RETURN ls_ret

```

## Use the undocumented INDIRECT keyword

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0133.html>

INDIRECT declaration statement enables a function to be called indirectly by simply setting an instance or shared variable. With a simple INDIRECTed variable, we need to provide 2 functions. One to set the value and one to retrieve the value.

```
[instance variable]
public:
    INDIRECT string i_username {of_SetUsername(*value), of_GetUsername()}

private:
    string zis_username

[Powerscript functions]
function integer of_SetUsername(string as_username)

    IF NOT IsNull(as_username) THEN
        zis_username = upper(as_username)
    END IF
    RETURN 1

function String of_GetUsername()
    RETURN zis_username

[test code]

// the of_SetUsername() function is called by PB
i_username = "powerbuilder howto"

// the of_GetUsername() function is called by PB
MessageBox("username", i_username)
```

Here some notes from Jeremy Lakeman (thanks!).

- INDIRECT variables only compile with 2 or 6 functions in the array otherwise the compiler will crash.
- Arguments that start with a \* will be replaced by the appropriate value at compile time and all arguments are optional, the compiler wont complain if you don't pass one of the valid values
- Arguments can be supplied in any order
- The compiler will crash if you define a function with an unsupported argument
- Extra identifiers can be passed to each method, but only if they are valid in the context of the calling code
- The argument names are :

- string \*name            the name of the variable (used with INDIRECT array)
- any        \*value        the value being assigned
- long[] \*args          the array of array dimensions (used with INDIRECT array)
- Where a value is supplied as an argument, the compiler will call any matching method. This means that you could override the set methods to allow multiple types of values to be assigned.

INDIRECT keyword when used with an array is a little bit more tricky, we need to define 6 functions.

In the following example, the INDIRECT array converts its member to uppercase. A userobject is used to hold our INDIRECT array and the 6 required functions.

[EXPORT .sru file, save into a file *n\_cst\_indirect.sru* and then IMPORT it in PB]

```
$PBExportHeader$n_cst_indirect.sru
forward
global type n_cst_indirect from nonvisualobject
end type
end forward

global type n_cst_indirect from nonvisualobject
end type
global n_cst_indirect n_cst_indirect

type variables

indirect string x[] {&
    of_set_array(*name, *value), &
    of_set_item(*name, *args, *value), &
    of_get_array(*name), &
    of_get_item(*name, *args), &
    of_upperbound(), &
    of_lowerbound() }

private:
string ix[]

end variables
forward prototypes
public function integer of_lowerbound ()
public function integer of_upperbound ()
private function any of_get_array (string s)
private function string of_get_item &

    (string s, long al_dimensions[])
private subroutine of_set_item &
```

```

(string s, long al_dimensions[], string as_value)
private subroutine of_set_array (string s, string as_values[])
end prototypes

public function integer of_lowerbound ();
// provide the lower bound of the array
// can be defined with a *value argument
// but then must be called by lowerbound(x, 1)
// this method must be public

return lowerbound(ix)
end function

public function integer of_upperbound ();
// provide the upper bound of the array
// can be defined with a *value argument
// but then must be called by upperbound(x, 1)
// this method must be public

return upperbound(ix)
end function

private function any of_get_array (string s);
// return the entire array in an any variable
any la_ret

la_ret=ix
return la_ret
end function

private function string of_get_item &

(string s, long al_dimensions[]);
// get an element of the array
return ix[al_dimensions[1]]
end function

private subroutine of_set_item &

(string s, long al_dimensions[], string as_value);
// set an element of the array
ix[al_dimensions[1]]=upper(as_value)
end subroutine

private subroutine of_set_array (string s, string as_values[]);
integer i
String ls_blank[]

ix = ls_blank
// set the entire array
FOR i = 1 TO upperbound(as_values)
  x[i] = as_values[i]
NEXT
end subroutine

```

```

on n_cst_indirect.create
call super::create
TriggerEvent( this, "constructor" )
end on

on n_cst_indirect.destroy
TriggerEvent( this, "destructor" )
call super::destroy
end on

```

To use the userobject from Powerscript :

```

integer i
n_cst_indirect lnv_i
String tmp[] = &

{ "the uppercase conversion", &
  "was done with the", &
  "INDIRECT keyword!"}

lnv_i = create n_cst_indirect
lnv_i.x[1] = "powerbuilder howto"
MessageBox("", lnv_i.x[1])

lnv_i.x[2] = "http://www.rgagnon.com"
MessageBox("", lnv_i.x[2])

Messagebox("", string(upperbound(lnv_i.x)))

lnv_i.x = tmp
FOR i = 1 TO Upperbound(lnv_i.x)
  MessageBox("", lnv_i.x[i])
NEXT

```

The same technique but this time, our array type is long.

```

$PBExportHeader$n_cst_indirect.sru
forward
global type n_cst_indirect from nonvisualobject
end type
end forward

global type n_cst_indirect from nonvisualobject
end type
global n_cst_indirect n_cst_indirect

type variables

indirect long x[] {&

```

```

of_set_array(*name, *value), &
of_set_item(*name, *args, *value), &
of_get_array(*name), &
of_get_item(*name, *args), &
of_upperbound(), &
of_lowerbound()}

private:
long ix[]

end variables

forward prototypes
public function integer of_lowerbound ()
public function integer of_upperbound ()
private function any of_get_array (string s)
private subroutine of_set_array (string s, long al_values[])
private function long of_get_item (string s, long al_dimensions[])
private subroutine of_set_item &
(string s, long al_dimensions[], long al_value)
end prototypes

public function integer of_lowerbound ();
return lowerbound(ix)
end function

public function integer of_upperbound ();
return upperbound(ix)
end function

private function any of_get_array (string s);
any la_ret

la_ret=ix
return la_ret
end function

private subroutine of_set_array (string s, long al_values[]);
ix=al_values
end subroutine

private function long of_get_item (string s, long al_dimensions[]);
return ix[al_dimensions[1]]
end function

private subroutine of_set_item &
(string s, long al_dimensions[], long al_value);

```

```

ix[al_dimensions[1]]=al_value
end subroutine

on n_cst_indirect.create
call super::create
TriggerEvent( this, "constructor" )
end on

on n_cst_indirect.destroy
TriggerEvent( this, "destructor" )
call super::destroy
end on

```

Array with multi-dimensions needs functions with different signatures :

```

$PBExportHeader$n_cst_indirect.sru
forward
global type n_cst_indirect from nonvisualobject
end type
end forward

global type n_cst_indirect from nonvisualobject
end type
global n_cst_indirect n_cst_indirect

type variables

indirect long x[1,2,3] {&

    of_set_array(*name, *value, *eoseq), &
    of_set_item(*name, *nargs, *args, *value, *eoseq), &
    of_get_array(*name, *eoseq), &
    of_get_item(*name, *nargs, *args, *eoseq), &
    of_upperbound(*dims), &
    of_lowerbound(*dims) }

private:
long ix[]

/*
indirect variables only compile with 2 or 6 functions in the array
otherwise the compiler will crash

arguments that start with a * will be replaced by the appropriate
value at compile time
all arguments are optional, the compiler wont complain if you don't
expect one of the valid values
arguments can be supplied in any order
the compiler will crash if you define a function with an unsuported

```

```

argument
extra identifiers can be passed to each method, but only if they are
valid in the context of the calling code
Eg you could pass "this" as an argument

string *name    the name of the variable
any   *value    the value being assigned
long[] *args    the array of array dimensions
long   *nargs   the number of elements in *args
boolean *eoseq  end of sequence
                  used to indicate if this is the end of a dot notation
                  sequence
integer *dims   dimension for lower or upper bound?
                  the compiler doesn't seem to care which named value
                  you actually use though

```

Where a value is supplied as an argument, the compiler will call any matching method. This means that you could override the set methods to allow multiple types of values to be assigned.

```
*/
```

```

end variables
forward prototypes
public function integer of_upperbound (integer al_dims)
public function integer of_lowerbound (integer al_dims)
private subroutine of_set_array &

(string as_name, long al_values[], boolean ab_eoseq)
private function any of_get_array (string as_name, boolean ab_eoseq)
private function long of_get_item &

(string as_name, long al_n_dimensions, &
     long al_dimensions[], boolean ab_eo_seq)
private subroutine of_set_item &

(string as_name, long al_n_dimensions, &
     long al_dimensions[], long al_value, boolean ab_eoseq)
end prototypes

public function integer of_upperbound (integer al_dims);
// provide the upper bound of the array
// can be defined with a * argument doesn't seem to care which one
// but then must be called by upperbound(x, 1)
// this method must be public

return upperbound(ix)
end function

public function integer of_lowerbound (integer al_dims);
// provide the lower bound of the array
// can be defined with a * argument doesn't seem to care which one
// but then must be called by lowerbound(x, 1)

```

```

// this method must be public

return lowerbound(ix)
end function

private subroutine of_set_array &

(string as_name, long al_values[], boolean ab_eoseq);
// set the entire array
ix=al_values
end subroutine

private function any of_get_array (string as_name, boolean ab_eoseq);
// return the entire array in an any variable
any la_ret

la_ret=ix
return la_ret
end function

private function long of_get_item &

(string as_name, long al_n_dimensions, long al_dimensions[], &
boolean ab_eo_seq);
// get an element of the array
return ix[al_dimensions[1]]
end function

private subroutine of_set_item &

(string as_name, long al_n_dimensions, long al_dimensions[], &
long al_value, boolean ab_eoseq);
// set an element of the array
ix[al_dimensions[1]]=al_value
end subroutine

on n_cst_indirect.create
call super::create
TriggerEvent( this, "constructor" )
end on

on n_cst_indirect.destroy
TriggerEvent( this, "destructor" )
call super::destroy
end on

```

## Encrypt/Decrypt a string

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0135.html>

```

[function string of_encrypt(as_str)]
integer i, j
string ls_enctext = ""
CONSTANT string CRYPT_KEY = "$#@%&%@&*"

j = len(as_str)
FOR i = 1 TO j
    ls_enctext += mid(CRYPT_KEY , mod(i,10) + 1, 1)
    ls_enctext += String(Char(255 - Asc(Mid(as_str, i, 1))))
NEXT

RETURN ls_enctext

[function string of_decrypt(as_str)]
integer i, j
string ls_encchar, ls_temp, ls_unasstr = "*** Encryption Error"
boolean lb_ok = true
CONSTANT string CRYPT_KEY = "$#@%&%@&*"

j = len(as_str)

IF NOT Mod(j, 2) = 1 THEN
    ls_temp = ""
    FOR i = 2 TO (j + 1) STEP 2
        ls_encchar = Mid(as_str, i - 1, 1)
        IF mid(CRYPT_KEY, Mod(i / 2, 10) + 1, 1) <> ls_encchar THEN
            lb_ok = FALSE
            EXIT
        END IF
        ls_encchar = Mid(as_str, i, 1)
        ls_temp += string(char(255 - asc(ls_encchar)))
    NEXT
END IF

IF lb_ok THEN ls_unasstr = ls_temp

RETURN LS_UNASSTR

```

Check this [How-to](#) for a solution using the Microsoft API.

## Check if an event is supported by an object

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0153.html>

```

any la_rc
integer li_rc

```

```

la_rc = my_uo.event dynamic ue_someevent()
IF IsNull(la_rc) THEN
    // action if event does not exist
ELSE
    li_rc = la_rc
    // action based on return value
END IF

```

## Using PFC

```

n_cst_metaclass lnv_metaclass
IF lnv_metaclass.of_IsEventDefined(my_uo.ClassName(), 'ue_someevent') THEN
    li_rc = my_uo.event dynamic ue_someevent()
    // action based on return value
ELSE
    // action if event not defined
END IF

```

## Check if an object definition exists

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0178.html>

For user object or window

```

ClassDefinition cd_dwo

cd_dwo = FindClassDefinition(as_objectname2check)

IF IsNull(cd_dwo) THEN
    MessageBox("Error", as_objectname2check &
        + " is not present in the present Library List!")
END IF

```

For datawindow

```

datastore lds_temp

lds_temp = CREATE datastore
lds_temp.dataobject = as_objectname2check

IF NOT IsValid(lds_temp.Object) THEN
    MessageBox("Error", "The datawindow " &
        + as_objectname2check + " does not exist!")
END IF

```

In PB8, the new try/catch can be used to detect missing objects.

```
TRY
open (w_missing_window)
CATCH (runtimeerror EX)
    MessageBox ("Error", "Window missing....")
    RETURN -1
END TRY
```

## Have a button with many lines

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0195.html>

Use *PictureButton* with no picture.

## Display Powerbuilder's easter egg

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0207.html>

Display the About... dialog and then type

PB7 - powersoft

PB8 - morepower

PB9 - 4glplus (must be 1024x800)

For more easter eggs, check this [site](#).

## Close a window with ESC

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0211.html>

Add the code below to the key event of the window:

```
CHOOSE CASE key
    CASE keyescape!
        close (this)
    END CHOOSE
```

## Terminate an another application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0215.html>

We post the WM\_QUIT message to an application (developped with Powerbuilder or not) to close it.

In this example, if the Windows Calculator is running then it will be closed from Powerscript.

The target window handle is retrieved by looking at its title.

```
[local external function declaration]
FUNCTION ulong FindWindowA(ulong classname, String windowname) &
    LIBRARY "user32.dll"
FUNCTION boolean PostMessageA&

(ulong hwndle,UINT wmsg,ulong wParam,ulong lParam) &
LIBRARY "user32.dll"

[powerscript]
CONSTANT uint WM_QUIT = 18 // hex 0x0012
ulong      lul_handle
string     ls_app

ls_app = "Calculator"
// ls_app = "Calculatrice"  in french windows!

lul_handle = FindWindowA(0, ls_app)

IF lul_handle > 0 THEN
    PostMessageA(lul_handle, WM_QUIT, 0, 0);
ELSE
    MessageBox("Oups", ls_app + " is not running!")
END IF
```

## Get the current function/function name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0219.html>

```
// dummy error code and error text
PopulateError(999, "notused")

Messagebox(error.object , error.objectevent)
//
```

```
// to trigger a SystemError : SignalError()
//
```

## Send email with attachment (MAPI)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0220.html>

```
mailsession lms_sess
mailmessage lm_mess
mailrecipient lm_recipient
mailFileDescription lmf_file
mailreturncode lmc_rc

lmf_file.FileType = mailAttach!
lmf_file.FileName = "report.psr"
lmf_file.PathName = "c:\temp\" + "report.psr"

lm_mess.NoteText = "this my report."
lm_mess.Subject = "report may2004"
lm_mess.Attachmentfile[1] = lmf_file

lm_recipient.Address = "someone@somewhere.com"
lm_recipient.Name = "Mr. Brown"
lm_recipient.RecipientType = mailTo!
lm_mess.Recipient[1] = lm_recipient

lms_sess = create mailsession
lmc_rc = lms_Session.mailLogon()
lmc_rc = lms_Session.mailSend(lm_Message)
IF lmc_rc <> mailReturnSuccess! THEN
    MessageBox("Error", "Error on Send")
END IF

lmc_rc = lms_sess.mailLogoff()
DESTROY lms_sess
```

## Use a Java class (PB9)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0224.html>

We simply need to build an "EJB proxy" to our class even if it's not an EJB.

Consider the following simple class :

```
[Simple.java]
public class Simple {
    public Simple() { }
    public static String sayHello(){
        return "Hello world";
    }
    public String sayNonStaticHello(){
        return "Hello world";
    }
    public static void main(String[] args) {
        System.out.println(Simple.sayHello());
    }
}
```

Before generating the proxy, we need a "dummy" SimpleHome class.

```
[SimpleHome.java]
public interface SimpleHome { }
```

Compile both files. Now, we will build a Powerbuilder application and use our "Simple" class from Powerscript.

1. Start Powerbuilder.
2. Create a new application, call it "pbjsimple", put everything in the same directory(.class and .pbl).
3. Start the EJB Proxies Wizard, Type "Simple" as the component and the classpath is the folder containing the "Simple.class" file.
4. Run the project, Powerbuilder will generate proxies for your classes and also for most standard Java classes!
5. Next, open a window from the application and drop a Button and insert the following Powerscript in the "click event".

```
JavaVM lJavaVM
EJBConnection lEJBConn
Simple lInv_simple
long ll_return

lJavaVM = CREATE JavaVM

// need to specify the classpath form the Simple class
ll_return = &

lJavaVM.CreateJavaVM("C:\Applications\regal\dev\Work\pb9\Target3", FALSE)
CHOOSE CASE ll_return
CASE 1
CASE 0
CASE -1
    MessageBox ( "", "jvm.dll was not found in the classpath." )
CASE -2
    MessageBox ( "", "pbejbclient90.jar file was not found." )
CASE ELSE
    MessageBox ( "", "Unknown result (" + String (ll_return ) + ")" )
END CHOOSE
```

```

IF ll_return <0 THEN
    DESTROY lJavaVM
END IF

lEJBConn = CREATE EJBConnection
ll_return = lEJBConn.CreateJavaInstance( lnv_simple, "Simple")
IF ll_return  0 THEN
    MessageBox("", "CreateJavaInstance returned " +string(ll_return))
    destroy lEJBConn
ELSE
TRY
    MessageBox("From Java!", lnv_simple.sayNonStaticHello())
CATCH (CreateException ce)
    MessageBox( "Create Exception", ce.getMessage() )
CATCH (Throwable t)
    MessageBox(" Other Exception", t.getMessage())
END TRY
END IF

```

Note : AFAIK Java static methods are not visible from Powerbuilder.

You can download this example [here](#)

## Know more about the Java support in PB9

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0225.html>

Actually the best place is in the Powerbuilder's HTML book "Application technique, chapter 39 - Java support".

### Locating the Java VM

PB uses an algorithm based on the location of the pbjvm90.dll file location to find the jvm.dll (the Java Virtual Machine). If not found, PowerBuilder uses the first jvm.dll whose location is defined in the user's PATH environment variable. If no jvm.dll is found, the Java VM does not start. It's not a bad idea to use the AppPath registry entry of your application instead of the system path definition.

### The runtime Java VM

You can create a special registry key (HKEY\_LOCAL\_MACHINE\Software\Sybase\PowerBuilder\9.0\Java) to override some JVM properties, this can be useful to specify a special "file.encoding" for example.

## Check if the current user belongs a specific Windows group/role

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0227.html>

You can connect to the ActiveDirectory or use some API but an easy way is to use a connection to SQL Server. Send the query :

```
SELECT is_member('mydomain\g_dept')
```

and the result can be

```
0      Current user is not a member of group or role.  
1      Current user is a member of group or role.  
NULL  Either group or role is not valid.
```

## Enable/disable a component from its name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0240.html>

Scan the Window Control array to extract a reference to the desired component. In this example, we are looking for a Command Button called "cb\_1".

```
Commandbutton lcb  
int i  
int j  
  
j = upperbound(parent.Control)  
FOR i = 1 TO j  
  IF parent.Control[i].TypeOf() = commandbutton! THEN  
    IF parent.Control[i].ClassName() = "cb_1" THEN  
      // we got it!  
      lcb = parent.Control[i]  
      EXIT  
    END IF  
  END IF  
NEXT  
  
IF NOT IsNull(lcb) THEN  
  lcb.enabled = false  
  lcb.text = "DISABLED"  
END IF
```

## Encode and display character using hex code

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0243.html>

The tilde is a special character in Powerbuilder used as an escape character in a string.

```
// prints copyright sign and "A" "B" using hex code (ansi)
String ls_string = "This a test ~hA9 ~h41 ~h42"
Messagebox("hex", ls_string)

// prints copyright sign and "A" "B" using decimal code (ansi)
ls_string = "This a test ~169 ~065 ~066"
Messagebox("decimal", ls_string)

// prints a tilde
ls_string = "This a test ~~"
Messagebox("tilde", ls_string)
```

## Use the clipboard

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0251.html>

Extract data from the clipboard

```
String ls_value
ls_value = Clipboard()
```

Put data into the clipboard

```
Clipboard("Hello")
```

A special attention is needed when calling Clipboard in a DataWindow control or DataStore object. To retrieve or replace the contents of the system clipboard with text from a DataWindow item (cell value), you must first assign the value to a string and then call the system Clipboard function as follows:

```
string ls_data = dw_1.object.column_name[row_number]
::Clipboard(ls_data)
```

That's because the DataWindow has its own Clipboard function and it is only applicable to graphs.

To store bitmap into the clipboard, you need to use some Windows API. See [this article](#)

See this [related howto](#)

See also this [related howto](#)

## Overload a global function

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0257.html>

thanks to Lawrence Sims for the following tip

You can overload global function objects in PowerBuilder with a simple trick. Just always edit them in source view, not in the painter. The following function (f\_log) can be called like so:

```
f_log("Message To Be Logged")
f_log(t_throwable_caught, populateError(0, ""))
f_log(populateError(0, "Got to here ..."))
```

[f\_log.srf]

just imported this file into a PBL to be able to use the overloaded f\_log function)

```
global type f_log from function_object
end type

type prototypes
subroutine OutputDebugString (string as_msg) library "kernel32.dll" &
    alias for "OutputDebugStringA"
end prototypes

forward prototypes
global subroutine f_log (readonly string as_msg)
global subroutine f_log (throwable at, readonly integer ai_nu)
global subroutine f_log (readonly integer ai_nu)
end prototypes

global subroutine f_log (readonly string as_msg);
OutputDebugString (as_msg)
end subroutine

global subroutine f_log (throwable at, readonly integer ai_nu);
string ls_message

ls_message = error.text
if isNull (error.text) or error.text = "" &
    then ls_message = at.getMessage ()

OutputDebugString (error.object + "." + error.objectevent + &
    ": line " + string (error.line) + ":" + ls_message)
end subroutine

global subroutine f_log (readonly integer ai_nu);
```

```

if isValid (error) then
    OutputDebugString (error.object + "." + error.objectevent + &
                       ": line " + string (error.line) + ":" + error.text)
end if
end subroutine

```

## Override system functions

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0261.html>

(thanks to J.Lakeman for this HowTo)

Let's say you want to override the behaviour of a system function, but you still want the option of calling it. For example you might have a client application that can also run as a service, so you might want calls to messagebox to be logged to a file instead of displayed.

In PB5 you could create your own instance of systemfunctions and call that. But since PB6 this is no longer possible. Powerbuilder decides which method you are calling at compile time.

For example, if you wrote a new messagebox method it won't be called by existing objects until your application is rebuilt. So what we need is an object that has compiled in calls to the real system functions, that won't get recompiled when we rebuild the application.

First up, create a new target application. Then build an nvo with methods that call the system functions. First up, create a new target application. Then build an nvo with methods that call the system functions. eg:

```

forward
global type n_cst_systemfuncs from nonvisualobject
end type
end forward

global type n_cst_systemfuncs from nonvisualobject autoinstantiate
end type

forward prototypes
public function integer of_messagebox(string c1, string t2)
end prototypes

public function integer of_messagebox(string c1, string t2);
    RETURN messagebox(c1, t2)
end function

on n_cst_systemfuncs.create
call super::create
TriggerEvent( this, "constructor" )
end on

```

```

on n_cst_systemfuncs.destroy
TriggerEvent( this, "destructor" )
call super::destroy
end on

```

You can generate the syntax for all the built in global functions from the class definition of the real systemfunctions object. Excluding any methods with a variable number of arguments.

Build this nvo into a PBD. And add it to the library list of your application.

Now you've got a pointer to the standard system function that you can call even when you have overriden it.

```

global type messagebox from function_object
end type

forward prototypes
global function integer messagebox (string as_caption, string as_text)
end prototypes

global function integer messagebox &

(string as_caption, string as_text);
n_cst_systemfuncs ln_funcs
IF gb_service THEN
  f_log(as_text)
  RETURN 1
ELSE
  RETURN ln_funcs.of_messagebox(as_caption, as_text)
END IF
end function

```

## Unicode String versus ANSI String (PB10)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0263.html>

Since PB is now Unicode-based, you may have a problem calling an external API which is expecting an ANSI string. The good news is that PB provides a way to convert the String to the required format.

```

Blob lbl_data

lbl_data = Blob("Hello World!", EncodingANSI!)
ls_data = String(lbl_data, EncodingANSI!)
...

```

The default PB10 encoding is UTF-16, little endian (EncodingUTF16LE!)

## STRINGS

In PB10, the PowerScript functions Len(), Left(), Mid(), and Right() are all Unicode character based and are equivalent to the existing LenW(), LeftW(), MidW(), and RightW() functions.

To manipulate strings as bytes or ASCII characters instead of Unicode characters, a new set of LenA(), LeftA(), MidA(), and RightA() functions have been added. When the 'A' functions are applied, PowerBuilder will convert the Unicode string to a DBCS string based on the machine's locale, and then apply the operation.

## FILES

The FileOpen() function now has an Encoding argument for identifying the encoding of the target file. The function FileEncoding(filename) determines and returns the encoding (EncodingANSI!, Encoding UTF8! , EncodingUTF16LE!, And EncodingUTF16BE!) used in the file. The FileEncoding() function should precede the opening of existing files so the correct encoding parameter can be pass in the FileOpen() call.

```
// PB10
li_FileNum = FileOpen("Employee.txt", TextMode!, Read!, EncodingANSI!)
// PB9
//li_FileNum = FileOpen("Employee.txt", TextMode!)
```

More details at [http://www.sybase.com/content/1036154/L02684\\_PB\\_InternationalizedApps\\_WP.pdf](http://www.sybase.com/content/1036154/L02684_PB_InternationalizedApps_WP.pdf)

## Remove accented letters from a String

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0264.html>

The following snippet removes from a String accented letters and replace them by their regular ASCII equivalent.

This can be useful before inserting data into a database to made sorting easier.

```
String accent    = "ÈÉÊËÛÙÏÎÀÂÔÖÇèéêûùïîàâôöç"
String noaccent = "EEEEUUUIIAAOOCeeeeuuiiaaooc"

String test = "Test : à é À É ç"
String currentChar = ""
String result = ""
int i,j = 0

FOR i = 1 TO len(test)
    currentChar = mid(test,i, 1)
    j = pos(accent, currentChar)
    IF j > 0 THEN
        result += mid(noaccent,j,1)
    ELSE
        result += currentChar
```

```

    END IF
NEXT

MessageBox(test, result)

/*
result :
-----
Test : à é À É ç
-----
Test : a e A E c
-----
OK
-----
*/

```

The weakness of the above HowTo is a potentially poor performance when dealing with a large string or many strings. The multiple concatenations of the result string will create many temporary strings. It's ok if you are dealing few lines but it's possible to do better.

The next version uses a blob to hold the result string. This version is more optimized since we are not creating any temporary string.

```

[function string of_removeaccent(string readonly as Accent)]
CONSTANT String ACCENT_LIST = "ÈÉËÜÙÏÌÀÂÔÖÇèéëûùïìàâôöç"
CONSTANT String NOACCENT_LIST = "EEEEUUUIIAAOOCeeeeuuiiaaooc"

CONSTANT int ENDOFSTRING_LEN = 2 // end of string PB10 (unicode)
//CONSTANT int ENDOFSTRING_LEN = 1 // end of string PB9 or less (ansi)

blob    result
string  currentChar, currentNoAccent
int     index, posAccent

ulong currentBlobIndex = 1

IF IsNull(as Accent)      THEN
    RETURN as Accent
END IF

IF Trim(as Accent) = "" THEN
    RETURN as Accent
END IF

// create a blob and add an extra character to take into account the End-of-string
result = Blob(as Accent + " ")

// scan the input
FOR index = 1 TO len(as Accent)
    currentChar = mid(as Accent, index , 1)

```

```

posAccent = pos(ACCENT_LIST, currentChar)
IF posAccent > 0 THEN
    currentNoAccent = mid(NOACCENT_LIST, posAccent, 1)
    currentBlobIndex= BlobEdit(result, currentBlobIndex, currentNoAccent)
ELSE
    currentBlobIndex= BlobEdit(result, currentBlobIndex, currentChar)
END IF
currentBlobIndex == ENDOFSTRING_LEN
NEXT

RETURN String(result)

[powerscript]
String test = "Test : à é à É ç"
String ls_result

ls_result = of_removeaccent(ls_test)
MessageBox(ls_test, ls_result)

```

---

See also this [Java HowTo](#)

## Get the execution path of the current application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0014.html>

```

[PB external function declaration]
FUNCTION int GetModuleFileNameA(&

    ulong hinstModule, &

    REF string lpszPath, &

    ulong cchPath) LIBRARY "kernel32"

[Powerscript]
string ls_Path
unsignedlong lul_Handle

ls_Path = space(1024)

lul_Handle = Handle(GetApplication())
GetModuleFileNameA(lul_Handle, ls_Path, 1024)
MessageBox("CURRENT APPLICATION PATH", ls_Path)

```

## Get the current directory

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0003.html>

First declare

```
FUNCTION long GetCurrentDirectoryA( long length , ref string path) &
LIBRARY "KERNEL32"
```

and then

```
long ll_ret
string ls_path

ls_path = Space(250)
ll_ret = GetCurrentDirectoryA(250, ls_path)
IF ll_ret > 0 THEN
    ls_path = Left(ls_path,ll_ret)
    MessageBoxBox("", ls_path)
ELSE
    Messagebox("Error", "Err GetCurrentDirectory " + String(ll_ret))
END IF
```

## Change directory

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0034.html>

```
[FUNCTION DECLARATIONS]
FUNCTION boolean SetCurrentDirectoryA(ref string lpsdir) &
LIBRARY "kernel32.dll"

[powerscript]
String ls_Directory

ls_Directory = "C:\MyNewDirectory\"

lb_Return = SetCurrentDirectoryA(ls_Directory)
```

If you're using the PFC, look at of\_ChangeDirectory in pfc\_n\_cst\_filesv.

## Create or remove a directory

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0004.html>

Declare these functions :

```
FUNCTION boolean CreateDirectoryA(ref string path, long attr)
    LIBRARY "kernel32.dll"
FUNCTION boolean RemoveDirectoryA( ref string path )
LIBRARY "KERNEL32.DLL"
```

and then

```
CreateDirectoryA( "C:\TempDir", 0 ) // always 0
RemoveDirectoryA( "c:\tempdir" )
```

You can't create more than one directory at a time. To create c:\TempDir\t1, you create \TempDir and then \TempDir\t1.

## Rename a file

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0005.html>

Simply "move" it under a new name with the function

```
[local function declaration]
FUNCTION BOOLEAN MoveFileA(STRING oldfile, STRING newfile) &
LIBRARY "KERNEL32.DLL"
```

Remember that PB10.5 is Unicode so you need to add the ;ANSI suffix to the declaration,

```
[local function declaration]
FUNCTION BOOLEAN MoveFileA(STRING oldfile, STRING newfile) &
LIBRARY "KERNEL32.DLL" ALIAS FOR "MoveFileA;ansi"
```

## Generate a unique filename

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0062.html>

```
[function declaration]
FUNCTION integer GetTempFileNameA  &

(ref string tempdir, ref string prefix, &
integer seed, ref string tempfile )&

LIBRARY "kernel32"

[powerscript]
integer li_rc
string ls_tempdir = "c:\temp"
string ls_prefixe = "app"
integer li_seed = 0
string ls_filename

ls_filename = space(256)
li_rc = GetTempFileNameA(ls_tempdir, ls_prefixe, li_seed, ls_filename)
IF li_rc = 0 THEN
    MessageBox("Oups", "Error")
ELSE
    MessageBox("Unique filename", ls_tempfile)
END IF
```

## Determine the TEMP directory designated for temporary files.

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0043.html>

In PB6, simply get the value of the environment variable *TEMP*

```
ContextKeyword lcxk_base
string ls_temp[]

this.GetContextService("Keyword", lcxk_base)
lcxk_base.getContextKeywords("TEMP", ls_temp)
RETURN ls_temp[1]
```

Or you can use the following API call

```
[External function declaration]
FUNCTION ulong GetTempPath(ulong nBufferLength, ref string lpBuffer) &

LIBRARY "kernel32" ALIAS FOR GetTempPathA

[powerscript]
long ll_bufferlength = 256
string ls_tempDir

ls_tempDir = SPACE(ll_bufferLength)
```

```

IF GetTempPath(ll_bufferLength, ls_tempDir) = 0 THEN
    MessageBox("Temp dir", "not defined")
ELSE
    MessageBox("Temp dir", ls_tempDir)
END IF

```

The GetTempPath function gets the temporary file path as follows:

6. The path specified by the TMP environment variable.
7. The path specified by the TEMP environment variable, if TMP is not defined.
8. The current directory, if both TMP and TEMP are not defined.

## Check if a file is open from another process

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0030.html>

Try to call the WinAPI CreateFile to open a given file in exclusive mode.

First, declare the following Local External Function (PB10)

```

FUNCTION Long CreateFile(ref string lpszName, long fdwAccess, long fdwShareMode, long lpsa, &
long fdwCreate, long fdwAttrsAndFlags, long hTemplateFile) LIBRARY "Kernel32.dll" &
ALIAS FOR "CreateFileA;Ansi"
FUNCTION boolean CloseHandle (long file_hand) LIBRARY "KERNEL32.DLL"

```

then from Powerscript :

```

CONSTANT ulong GENERIC_ACCESS = 268435456 // &H10000000
CONSTANT ulong EXCLUSIVE_ACCESS = 0
CONSTANT ulong OPEN_EXISTING = 3

long ll_handle
String ls_file

ls_file = "c:\temp\myfile.xls"

ll_handle = CreateFile ( ls_file, GENERIC_ACCESS, EXCLUSIVE_ACCESS, 0, OPEN_EXISTING, 0, 0 )
IF ll_handle <1 THEN
    MessageBox("", "Can't open, maybe missing or already opened ?!?")
ELSE
    MessageBox("", "File can be opened")
END IF

CloseHandle(ll_handle)

```

## Read big file (>32765 bytes)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0056.html>

The FileRead function is limited to read only 32765 bytes in one shot. The solution is to read the file until there is no more data to be read.

```
blob lbl_data
blob lbl_temp
long ll_file

ll_file = fileopen("mybigfile.txt", streammode!)

DO WHILE FileRead(ll_file, lbl_temp) > 0
    lbl_data += lbl_temp
LOOP

FileClose(ll_file)
```

## Set File Attribute

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0134.html>

```
[external function declaration]
FUNCTION boolean SetFileAttributesA &

(string lpFileName, unsignedlong dwFileAttributes) &

LIBRARY "Kernel32.DLL"

[powerscript]
/*
#define FILE_ATTRIBUTE_READONLY          0x00000001
#define FILE_ATTRIBUTE_HIDDEN           0x00000002
#define FILE_ATTRIBUTE_SYSTEM           0x00000004
#define FILE_ATTRIBUTE_DIRECTORY        0x00000010
#define FILE_ATTRIBUTE_ARCHIVE          0x00000020
#define FILE_ATTRIBUTE_NORMAL           0x00000080
#define FILE_ATTRIBUTE_TEMPORARY        0x00000100
#define FILE_ATTRIBUTE_COMPRESSED       0x00000800
#define FILE_ATTRIBUTE_OFFLINE          0x00001000
*/
CONSTANT unsignedlong FILE_ATTRIBUTE_READONLY = 1
```

```

CONSTANT unsignedlong FILE_ATTRIBUTE_HIDDEN = 2
CONSTANT unsignedlong FILE_ATTRIBUTE_SYSTEM = 4
CONSTANT unsignedlong FILE_ATTRIBUTE_ARCHIVE = 32
CONSTANT unsignedlong FILE_ATTRIBUTE_NORMAL = 128

IF NOT SetFileAttributesA("myfile.dat", FILE_ATTRIBUTE_READONLY) THEN
    MessageBox("Error", "can't set r/o attribute", Exclamation!)
END IF

```

## Browse for a folder using the standard Win dialog

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0047.html>

```

[External FUNCTION declaration]
FUNCTION long SHGetPathFromIDListA (long pidl, ref string pszPath) &

LIBRARY "shell32.dll"
FUNCTION long SHBrowseForFolderA ( BROWSEINFO lpBrowseInfo) &

LIBRARY "shell32.dll"

[structure BROWSEINFO]
howner      long
pidlroot    long
pszdisplayname string
lpsztitle   string
ulflags     long
lpfn        long
lparam      long
iimage      long

[Powerscript]
browseinfo lst_bi
long      ll_pidl
int       li_thispos
string    ls_path, ls_xs
long      ll_rc

ls_path=space(512)      // maximum path lenght
lst_bi.howner=handle(this) // handle of the window
lst_bi.pidlRoot = 0
lst_bi.pszdisplayname = space(260)
lst_bi.lpszTitle = "Select a folder"
lst_bi.ulFlags = 1
ll_pidl = SHBrowseForFolderA(lst_bi)
ll_rc = SHGetPathFromIDListA(ll_pidl, ls_path)
IF NOT ll_rc > 0 THEN
    SetNull(ls_path)

```

```
END IF
MessageBox("Folder selected" , ls_path)
```

NOTE: This may cause some memory leaks. To fix them, use the following API.

```
[external function declaration]
SUBROUTINE CoTaskMemFree( long lpv ) LIBRARY "ole32.dll"

[powerscript]
SHBrowseForFolder( lst_bi )
CoTaskMemFree( lst_bi.pidlRoot )
```

## Retrieve a file timestamp

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0205.html>

```
[structure]
type os_filedatetime from structure
  unsignedlong ul_lowdatetime
  unsignedlong ul_highdatetime
end type

type os_finddata from structure
  unsignedlong ul_fileattributes
  os_filedatetime str_creationtime
  os_filedatetime str_lastaccesstime
  os_filedatetime str_lastwritetime
  unsignedlong ul_filesizehigh
  unsignedlong ul_filesizelow
  unsignedlong ul_reserved0
  unsignedlong ul_reserved1
  character ch_filename[260]
  character ch_alternatefilename[14]
end type

type os_systemdatetime from structure
  uint      wYear
  uint      wMonth
  uint      wDayOfWeek
  uint      wDay
  uint      wHour
  uint      wMinute
  uint      wSecond
  uint      wMillisecond
end type

[external function declaration]
FUNCTION long FindFirstFileA &
  ( string filename, ref os_finddata findfiledata) &
```

```

LIBRARY "KERNEL32.DLL"
FUNCTION boolean FindClose (long handle) LIBRARY "KERNEL32.DLL"
FUNCTION boolean FileTimeToLocalFileTime &
    ( ref os_filedatetime lpFileTime, ref os_filedatetime lpLocalFileTime) &
    LIBRARY "KERNEL32.DLL"
FUNCTION boolean FileTimeToSystemTime &
    (ref os_filedatetime lpFileTime, ref os_systemdatetime lpSystemTime) &
    LIBRARY "KERNEL32.DLL"

[powerscript]
os_FindData lstr_FindData
os_FindData lstr_FindDataTemp
os_SystemDatetime lstr_SystemDateTime
long handle
String ls_timestamp

handle=FindFirstFileA &

("C:\Program Files\Sybase\PowerBuilder 9.0\pb90.exe", lstr_FindData)

IF handle==1 THEN
    // something wrong!
ELSE
    FindClose(handle)
    FileTimeToLocalFileTime &
        (lstr_FindData.str_lastwritetime, lstr_FindDataTemp.str_lastwritetime)
    IF FileTimeToSystemTime &
        (lstr_FindDataTemp.str_lastwritetime, lstr_SystemDateTime) THEN
        ls_timestamp = string &
            (datetime(date(lstr_SystemDateTime.wYear, &
                lstr_SystemDateTime.wMonth, lstr_SystemDateTime.wDay), &
                time(lstr_SystemDateTime.wHour, &
                    lstr_SystemDateTime.wMinute, lstr_SystemDateTime.wSecond)), &
                "mm/dd/yyyy hh:mm:ss")
        MessageBox("PB9", ls_timestamp)
    END IF
END IF

```

**ul\_fileattributes** in the **os\_finddata** contains the file attribute, the possible (hex) value are

FILE_ATTRIBUTE_READONLY	0x00000001
FILE_ATTRIBUTE_HIDDEN	0x00000002
FILE_ATTRIBUTE_SYSTEM	0x00000004
FILE_ATTRIBUTE_DIRECTORY	0x00000010
FILE_ATTRIBUTE_ARCHIVE	0x00000020
FILE_ATTRIBUTE_DEVICE	0x00000040
FILE_ATTRIBUTE_NORMAL	0x00000080
FILE_ATTRIBUTE_TEMPORARY	0x00000100
FILE_ATTRIBUTE_SPARSE_FILE	0x00000200
FILE_ATTRIBUTE_REPARSE_POINT	0x00000400
FILE_ATTRIBUTE_COMPRESSED	0x00000800
FILE_ATTRIBUTE_OFFLINE	0x00001000
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	0x00002000

```
FILE_ATTRIBUTE_ENCRYPTED          0x00004000
```

The value 0x00000021 is the result of FILE\_ATTRIBUTE\_READONLY + FILE\_ATTRIBUTE\_ARCHIVE.

## Convert a short pathname to a long

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0259.html>

The API GetLongPathName was introduced with Win98 and is unavailable on Win95 or WinNT but OK with XP.

```
[local external definition]
FUNCTION ulong GetLongPathName( &
    REF String lpszShortPath, &
    REF String lpszLongPath, &
    ULONG cchBuffer ) &

LIBRARY "kernel32.dll" ALIAS FOR "GetLongPathNameA"

[powerscript]
ContextKeyword lcxk_base
string ls_path_short
string ls_path_long
string ls_values[]

this.GetContextService("Keyword", lcxk_base)
lcxk_base.GetContextKeywords("temp", ls_values)
IF Upperbound(ls_values) > 0 THEN
    ls_path_short = ls_values[1]
ELSE
    ls_path_short = "*UNDEFINED*"
END IF

MessageBox("short",ls_path_short)

// ex : C:\DOCUME~1\Real\LOCALS~1\Temp

ls_path_long = space(255)

GetLongPathName(ls_path_short, ls_path_long, 255);

MessageBox("long",ls_path_long)

// ex : C:\Documents and Settings\Real\Local Settings\Temp
```

To do the conversion long to short, use GetShortPathName API. In PB10, you may need to use the Unicode version, so use the "ALIAS GetLongPathNameW" instead.

## Truncate a long path with ellipses

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0262.html>

The *PathCompactPathEx* truncates a path to fit within a certain number of characters by replacing path components with ellipses.

```
[local function declaration]
FUNCTION boolean PathCompactPathEx  &

( REF String pszOut, string szPath, uint cchMax, long dwFlags) &

ALIAS FOR "PathCompactPathExA" &

LIBRARY "shlwapi.dll"

[powerscript]
String ls_longPath = &

"C:\Applications\Netscape\Netscape8\defaults\profile\chrome"

// 30 chars max plus a null
String ls_truncatedPath = space(31)

PathCompactPathEx(ls_truncatedPath, ls_longPath, 31, 0)

MessageBox(ls_longPath, ls_truncatedPath)

/*
Output :
-----
C:\Applications\Netscape\Netscape8\defaults\profile\chrome
-----
C:\Applications\Nets...chrome
-----
OK
-----
*/
```

## Validate a date

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0029.html>

```
IsDate( String( DT_MYDATE, 'YYYY-MM-DD' ))
```

## Obtain the day of the year

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0031.html>

```
Date ld_today, ld_januaryFirst
int li_year
long ll_dayOfTheYear
ld_today = Today()
li_year = Year(ld_today)
ld_januaryFirst = Date( String( li_year ) + '-01-01' )
ll_dayOfTheYear = DaysAfter( ld_januaryFirst , ld_today ) + 1
```

## Obtain the week of the year

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0032.html>

```
// a week # (1-53)
Date ld_today, ld_januaryFirst
int li_year
long ll_weekOfTheYear
dt_today = Today()
li_year = Year(ld_today)
ld_januaryFirst = Date( String( li_year ) + '-01-01' )
ll_weekOfTheYear = Int( ( DaysAfter(ld_januaryFirst, Date(ld_date)) + &
                           DayNumber(ld_januaryFirst) ) / 7) + 1
```

## Set the system date/time

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0053.html>

```
[structure definition , str_systemtime]
year      uint
month     uint
dayweek   uint
```

```

day      uint
hour     uint
min      uint
sec      uint
millsec  uint

[local external function declaration]
FUNCTION long SetLocalTime(ref str_systemtime lpSystemTime ) &

LIBRARY "kernel32.dll"

[clicked event of a button]
str_systemtime lstr_systemtime

lstr_systemtime.year      = 1999
lstr_systemtime.month     = 1      // January = 1 and so on.
lstr_systemtime.dayweek   = 0      // not used
lstr_systemtime.day        = 3
lstr_systemtime.hour      = 12
lstr_systemtime.min       = 0
lstr_systemtime.sec       = 0
lstr_systemtime.millsec   = 0

SetLocalTime(lstr_systemtime)

```

## Obtain the last day of a month

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0058.html>

Take a date, increment the month, change the day for "01" then decrement by 1.

```

int li_retdays, li_month, li_year

li_month = Month(ad_date)
li_year = year(ad_date)

IF li_month <12 THEN
    li_month ++
ELSE
    li_month = 1
    li_year ++
END IF

// build a new date
ld_newdate = date(li_year,li_month,1)
// extract the last day of the previous month
ld_previousmonthlastday = Day(RealtiveDate(ld_newdate, -1))

```

## Get the time and date from a server

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0250.html>

First we need these local function declarations

```
[local function declaration]
FUNCTION ULONG GetTimeOfDayInfo &

    ( REF s_timeofday dest, ULONG source, ULONG size ) &
    LIBRARY "KERNEL32.DLL" ALIAS FOR "RtlMoveMemory"

FUNCTION ULONG NetRemoteTOD &

    (REF CHAR server[], REF ULONG bufferPtr) LIBRARY "NETAPI32.DLL"
FUNCTION ULONG LocalhostTOD &

    (ref long null, REF ULONG bufferPtr) LIBRARY "NETAPI32.DLL" &
    ALIAS FOR "NetRemoteTOD"
```

this structure

```
[s_timeofday structure]
elapsedt uLong
msecs uLong
hours uLong
mins uLong
secs uLong
hunds uLong
timezone Long
tinterval uLong
day uLong
month uLong
year uLong
weekday uLong
```

and these functions

```
[powerscript]
// -----
of_StringToUnicode (String as_string, ref character ac_unicode[])

int li_loop, li_len, li_uni

li_len = Len( as_string )
FOR li_loop = 1 TO li_len
    li_uni++
```

```

ac_unicode[ li_uni ] = Mid( as_string, li_loop, 1 )
li_uni++
ac_unicode[ li_uni ] = Char( 0 )
NEXT

li_uni++
ac_unicode[ li_uni ] = Char( 0 )
li_uni++
ac_unicode[ li_uni ] = Char( 0 )

// -----
datetime of_relativetimedate(datetime adt_start, long al_offset)

// stolen from the PFC!
datetime ldt_null
date ld_sdate
time lt_stime
long ll_date_adjust
long ll_time_adjust, ll_time_test

//Initialize date and time portion
ld_sdate = date(adt_start)
lt_stime = time(adt_start)

//Find out how many days are contained
//Note: 86400 is # of seconds in a day
ll_date_adjust = al_offset / 86400
ll_time_adjust = mod(al_offset, 86400)

//Adjust date portion
ld_sdate = RelativeDate(ld_sdate, ll_date_adjust)

//Adjust time portion
// Allow for time adjustments periods crossing over days
// Check for time rolling forwards a day
IF ll_time_adjust > 0 THEN
    ll_time_test = SecondsAfter(lt_stime,time('23:59:59'))
    IF ll_time_test < ll_time_adjust THEN
        ld_sdate = RelativeDate(ld_sdate,1)
        ll_time_adjust = ll_time_adjust - ll_time_test -1
        lt_stime = time('00:00:00')
    END IF
    lt_stime = RelativeTime(lt_stime, ll_time_adjust)
//Check for time rolling backwards a day
ELSEIF ll_time_adjust < 0 THEN
    ll_time_test = SecondsAfter(lt_stime,time('00:00:00'))
    IF ll_time_test > ll_time_adjust THEN
        ld_sdate = RelativeDate(ld_sdate,-1)
        ll_time_adjust = ll_time_adjust - ll_time_test +1
        lt_stime = time('23:59:59')
    END IF
    lt_stime = RelativeTime(lt_stime, ll_time_adjust)
END IF

```

```
RETURN (datetime(ld_sdate,lt_stime))
```

then the actual call to get the time and date from a server (or the localhost), we need to adjust the value returned because of the timezone offset.

[powerscript]

```
char lc_logonserver[]
string ls_logonserver
boolean lb_logonserver

ulong ul_prtTimeOfDayInfo
ulong ul_rc
long ll_null

s_timeofday lstr_timedate
ContextKeyword lcxk_base
String ls_result
string ls_values[]

datetime ldt_locale
date ld_server
time lt_server
datetime ldt_server

// get the logon server
this.GetContextService("Keyword", lcxk_base)
lcxk_base.GetContextKeywords("LOGONSERVER", ls_values)
IF Upperbound(ls_values) > 0 THEN
    ls_logonserver = ls_values[1]
    // transform logon server to unicode
    of_StringToUnicode(ls_logonserver, lc_logonserver)
    lb_logonserver = true
ELSE
    // lc_logonserver is null --> no server -> localhost
    lb_logonserver = false
END IF

// get the current time of day
IF lb_logonserver THEN
    ul_rc = NetRemoteTOD(lc_logonserver, ul_prtTimeOfDayInfo)
ELSE
    ll_null = 0
    ul_rc = LocalhostTOD(ll_null, ul_prtTimeOfDayInfo)
END IF
//

IF NOT ul_rc = 0 THEN
    MessageBox("Oops", String (ul_rc))
ELSE
    // convert to our structure
    GetTimeOfDayInfo(lstr_timedate, ul_prtTimeOfDayInfo, 48)
    ls_result = string(lstr_timedate.year) + "-" &
```

```

+ string(lstr_timedate.month) &
+ "—" + string(lstr_timedate.day) &
+ " " + string(lstr_timedate.hours) + ":" &
+ string(lstr_timedate.mins) &
+ ":" + string(lstr_timedate.secs)
MessageBox("result before conversion" , ls_result )

// convert to local time taking into account the timezone
ld_server = Date (string(lstr_timedate.year) + "-
+ string(lstr_timedate.month) + &
"-" + string(lstr_timedate.day))
lt_server = Time (string(lstr_timedate.hours) + ":" &
+ string(lstr_timedate.mins) + &
":" + string(lstr_timedate.secs))

ldt_locale = of_relativeDatetime &
(datetime(ld_server, lt_server), &
- (lstr_timedate.timezone * 60))
Messagebox("result after conversion" , &
String(ldt_locale, "yyyy-mm-dd hh:mm"))
END IF

```

See also this [HowTo](#)

---



---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>

## POWERBUILDER HOW-TO

# WinAPI/Registry

---

## pb-winapiregistry

---

### Get the network domain name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0125.html>

Here how you get the network username using the Windows Scripting Host :

```
OleObject wsh
Integer li_rc

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Network" )
IF li_rc = 0 THEN
    MessageBox( "Domain", String( wsh.UserDomain ) )
END IF
```

By calling WSH-VBScript functions, we can achieve some useful tasks very easily.

The next example shows you how to start Notepad and send some keys to it.

```
OleObject wsh
Integer li_rc

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Shell" )

wsh.Run ("Notepad")
Sleep (500)
wsh.AppActivate("Untitled - Notepad")
wsh.SendKeys ("hello from PB")
```

The declaration for the Sleep API is

```
[local external function declaration]
SUBROUTINE Sleep(Long lMilliSec) LIBRARY "Kernel32.dll"
```

NOTE: Recent version of WSH have their own Sleep function.

This one is calling the Windows Calculator

```
oleobject wsh
```

```

long ll_rc

wsh = CREATE oleobject
ll_rc = wsh.ConnectToNewObject ("WScript.Shell")
IF ll_rc <0 THEN
  messagebox("error","error")
END IF
wsh.Run( "calc")
Sleep (100)
wsh.AppActivate( "Calculator")
Sleep (100)
wsh.SendKeys( "1{+}")
Sleep (500)
wsh.SendKeys ("2")
Sleep (500)
wsh.SendKeys( "=")
Sleep (500)
wsh.SendKeys( "*4" )
Sleep (500)
wsh.SendKeys( "=" )
// 1+2 = 3 * 4 = 12

```

SendKeys can send "special key" using the following code :

```

BACKSPACE {BACKSPACE}, {BS}, or {BKSP}
BREAK {BREAK}
CAPS LOCK {CAPSLOCK}
DEL or DELETE {DELETE} or {DEL}
DOWN ARROW {DOWN}
END {END}
ENTER {ENTER} or ~
ESC {ESC}
HELP {HELP}
HOME {HOME}
INS or INSERT {INSERT} or {INS}
LEFT ARROW {LEFT}
NUM LOCK {NUMLOCK}
PAGE DOWN {PGDN}
PAGE UP {PGUP}
PRINT SCREEN {PRTSC}
RIGHT ARROW {RIGHT}
SCROLL LOCK {SCROLLOCK}
TAB {TAB}
UP ARROW {UP}
F1 {F1}
F2 {F2}
F3 {F3}
F4 {F4}
F5 {F5}
F6 {F6}
F7 {F7}
F8 {F8}
F9 {F9}
F10 {F10}

```

```
F11 {F11}
F12 {F12}
F13 {F13}
F14 {F14}
F15 {F15}
F16 {F16}
SHIFT +
CTRL ^
ALT %
```

You can use some vbscript to do things which can't be done easily in powerscript like binary operations or hexadecimal conversion :

```
OleObject wsh
integer li_rc, i, j, k
long m
string s

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"

i = 1
j = 2

k = integer(wsh.Eval( string(i) + " xor " + string(j)))

MessageBox( "Result" , string(i) + " xor " + string(2) + " = " + string(k))

// or CInt for integer
m = long(wsh.Eval( "CLng(~&HFF0F~)"))
MessageBox( "Result" , " HEX &HFF0F -> DEC " + string(m))

s = wsh.Eval( "hex(255)")
MessageBox( "Result" , " DEC 255 -> HEX " + s)

s = wsh.Eval( "hex(65000)")
MessageBox( "Result" , " DEC 65000 -> HEX " + s)
```

Call the Windows RUN dialog :

```
OleObject wsh

wsh = CREATE OleObject
wsh.ConnectToNewObject( "Shell.Application" )

wsh.filerun
```

You can even create some VBScript code on the fly with PB and execute it.

```
OleObject wsh
Integer li_rc, i, j, k
```

```
wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"
wsh.addcode("function retfnc(s)  retfnc=s  end function")
wsh.executestatement ('msgbox retfnc("true")')
```

See this [related howto](#)

## Use a Java object from PB

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0057.html>

See this [Java How-to](#) (using the Microsoft JVM).

See this [PB How-to](#) for a PB9 solution.

## Use WSH-VBScript functionalities

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0125.html>

Here how you get the network username using the Windows Scripting Host :

```
OleObject wsh
Integer li_rc

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Network" )
IF li_rc = 0 THEN
  MessageBox( "Domain", String( wsh.UserDomain ) )
END IF
```

By calling WSH-VBScript functions, we can achieve some useful tasks very easily.

The next example shows you how to start Notepad and send some keys to it.

```
OleObject wsh
Integer li_rc

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Shell" )

wsh.Run("Notepad")
```

```
Sleep(500)
wsh.AppActivate("Untitled - Notepad")
wsh.SendKeys("hello from PB")
```

The declaration for the Sleep API is

```
[local external function declaration]
SUBROUTINE Sleep(Long lMilliSec) LIBRARY "Kernel32.dll"
```

NOTE: Recent version of WSH have their own Sleep function.

This one is calling the Windows Calculator

```
oleoobject wsh
long ll_rc

wsh = CREATE oleoobject
ll_rc = wsh.ConnectToNewObject("WScript.Shell")
IF ll_rc <0 THEN
  messagebox("error","error")
END IF
wsh.Run( "calc")
Sleep (100)
wsh.AppActivate( "Calculator")
Sleep (100)
wsh.SendKeys( "1{+}")
Sleep (500)
wsh.SendKeys ("2")
Sleep (500)
wsh.SendKeys( "=")
Sleep (500)
wsh.SendKeys( "*4" )
Sleep (500)
wsh.SendKeys( "=" )
// 1+2 = 3 * 4 = 12
```

SendKeys can send "special key" using the following code :

```
BACKSPACE {BACKSPACE}, {BS}, or {BKSP}
BREAK {BREAK}
CAPS LOCK {CAPSLOCK}
DEL or DELETE {DELETE} or {DEL}
DOWN ARROW {DOWN}
END {END}
ENTER {ENTER} or ~
ESC {ESC}
HELP {HELP}
HOME {HOME}
INS or INSERT {INSERT} or {INS}
LEFT ARROW {LEFT}
NUM LOCK {NUMLOCK}
PAGE DOWN {PGDN}
PAGE UP {PGUP}
```

```

PRINT SCREEN {PRTSC}
RIGHT ARROW {RIGHT}
SCROLL LOCK {SCROLLLOCK}
TAB {TAB}
UP ARROW {UP}
F1 {F1}
F2 {F2}
F3 {F3}
F4 {F4}
F5 {F5}
F6 {F6}
F7 {F7}
F8 {F8}
F9 {F9}
F10 {F10}
F11 {F11}
F12 {F12}
F13 {F13}
F14 {F14}
F15 {F15}
F16 {F16}
SHIFT +
CTRL ^
ALT %

```

You can use some vbscript to do things which can't be done easily in powerscript like binary operations or hexadecimal conversion :

```

OleObject wsh
integer li_rc, i, j , k
long m
string s

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"

i = 1
j = 2

k = integer(wsh.Eval( string(i) + " xor " + string(j)))

MessageBox( "Result" , string(i) + " xor " + string(2) + " = " + string(k))

// or CInt for integer
m = long(wsh.Eval( "CLng(~&HFF0F~")"))
MessageBox( "Result" , " HEX &HFF0F -> DEC " + string(m))

s = wsh.Eval( "hex(255)" )
MessageBox( "Result" , " DEC 255 -> HEX " + s)

s = wsh.Eval( "hex(65000)" )
MessageBox( "Result" , " DEC 65000 -> HEX " + s)

```

Call the Windows RUN dialog :

```
OleObject wsh

wsh = CREATE OleObject
wsh.ConnectToNewObject( "Shell.Application" )

wsh.filerun
```

You can even create some VBScript code on the fly with PB and execute it.

```
OleObject wsh
Integer li_rc, i, j, k

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"
wsh.addcode("function retfnc(s)  retfnc=s  end function")
wsh.executestatement ('msgbox retfnc("true")')
```

See this [related howto](#)

## Check if a program is running

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0162.html>

```
OleObject locator,service,props
String ls_query = &

'select name , description from Win32_Process where name = "textpad.exe"'
int num, ret, i
locator = CREATE OleObject
ret = locator.ConnectToNewObject("WbemScripting.SWbemLocator");
service = locator.ConnectServer();
props = service.ExecQuery(ls_query);
num = props.count()

IF num > 0 THEN
    MessageBox("Process","textpad.exe is running")
ELSE
    Messagebox("Process","textpad.exe is NOT running")
END IF
```

Before using this technique, you may need to download and install the WMI ActiveX, here the [URL](#).

See this [related howto](#)

## Get information for the current user from ActiveDirectory

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0161.html>

```
OleObject ads

ads = CREATE OleObject
ads.ConnectToNewObject( "ADSystemInfo" )

// CN=Doe\,Joe,OU=something,OU=somewhere,DC=bigbuilding,DC=local
MessageBox( "", String( ads.UserName ) )
```

## Run a program and wait

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0151.html>

```
OleObject wsh
integer li_rc

CONSTANT integer MAXIMIZED = 3
CONSTANT integer MINIMIZED = 2
CONSTANT integer NORMAL = 1
CONSTANT boolean WAIT = TRUE
CONSTANT boolean NOWAIT = FALSE

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Shell" )

li_rc = wsh.Run("Notepad" , NORMAL, WAIT)
messagebox("hello", "world")
```

If WAITING, then Run returns any error code returned by the application.

In the example only three CONSTANTS are defined. But more options are available :

- 0 Hides the window and activates another window.
- 1 Activates and displays a window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when displaying the window for the first time.
- 2 Activates the window and displays it as a minimized window.
- 3 Activates the window and displays it as a maximized window.
- 4 Displays a window in its most recent size and position. The

```

active window remains active.
5 Activates the window and displays it in its current size and
position.
6 Minimizes the specified window and activates the next top-level
window in the Z order.
7 Displays the window as a minimized window. The active window
remains active.
8 Displays the window in its current state. The active window
remains active.
9 Activates and displays the window. If the window is minimized or
maximized, the system restores it to its original size and position. An
application should specify this flag when restoring a minimized window.
10 Sets the show state based on the state of the program that
started the application.

```

## Create a shortcut

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0182.html>

```

oleobject ws, loo_shortcut
string ls_desktop

ws = CREATE oleobject
ws.ConnectToNewObject("WScript.Shell")

// other "specialfolder" of interest
// AllUsersDesktop
// AllUsersStartMenu
// AllUsersPrograms
// AllUsersStartup
// Desktop
// Favorites
// MyDocuments
// Programs
// Recent
// SendTo
// StartMenu
// Startup
ls_desktop = ws.SpecialFolders("Desktop")
loo_shortcut = ws.CreateShortcut(ls_desktop +"\realhowto.lnk")
loo_shortcut.TargetPath = "http://www.rgagnon.com/howto.html"
loo_shortcut.WorkingDirectory = ls_desktop

// 1 Activates and displays a window.
// If the window is minimized or maximized, the system restores it to
// its original size and position.
// 3 Activates the window and displays it as a maximized window.
// 7 Minimizes the window and activates the next top-level window.
loo_shortcut.WindowStyle = 1

```

```
loo_shortcut.IconLocation = &  
    ws.ExpandEnvironmentStrings("%WINDIR%\SYSTEM\url.dll , 0")  
loo_shortcut.Save
```

CreateShortcut() is also used to read an existing shortcut.

```
ls_desktop = ws.SpecialFolders("AllUsersDesktop")  
loo_shortcut = ws.CreateShortcut(ls_desktop +"\\Mozilla Firefox.lnk")  
ls_firefoxPath = loo_shortcut.TargetPath  
  
MessageBox("FF path", ls_firefoxPath)
```

## Use XML

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0186.html>

The easiest way is to use the Microsoft XML parser (MSXML - a COM object) included with every recent Windows installation.

See [How to utilize XML technology with PowerBuilder](#) article on the mySybase site.

Also check <http://www.myjavaserver.com/~unoksoftgroup/xml/xml.html> to get a nice user object to handle XML data.

Note that Powerbuilder 9 will be able to handle XML files through datawindow and powerscript.

## Encode/decode URL's parameters

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0198.html>

Powerscript doesn't provide functions like Javascript's escape/unescape or Java's encode/decode.

The trick to use the Javascript engine from Powerscript.

```
OleObject wsh  
Integer li_rc  
string ls_temp  
  
wsh = CREATE OleObject  
li_rc = wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )  
wsh.language = "javascript"
```

```
ls_temp = wsh.Eval("escape('H e l l o')")  
  
MessageBox( "ESCAPE" , ls_temp)  
  
ls_temp = wsh.Eval("unescape('" + ls_temp + "')")  
  
MessageBox( "UNESCAPE" , ls_temp)
```

## Display PDF into a Window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0208.html>

Insert the "Adobe Acrobat ActiveX control" OLE control on the window.

Then to load a document from Powerscript

```
ole_1.object.LoadFile("C:\realhowto-pdf")
```

With a tool to explore the pdf.ocx, there are more functions :

```
function LoadFile(none fileName: WideString): WordBool [dispid $00000002]; stdcall;  
procedure setShowToolbar(none On: WordBool) [dispid $00000003]; stdcall;  
procedure gotoFirstPage [dispid $00000004]; stdcall;  
procedure gotoLastPage [dispid $00000005]; stdcall;  
procedure gotoNextPage [dispid $00000006]; stdcall;  
procedure gotoPreviousPage [dispid $00000007]; stdcall;  
procedure setCurrentPage(none n: Integer) [dispid $00000008]; stdcall;  
procedure goForwardStack [dispid $00000009]; stdcall;  
procedure goBackwardStack [dispid $0000000A]; stdcall;  
procedure setPageMode(none pageMode: WideString) [dispid $0000000B]; stdcall;  
procedure setLayoutMode(none layoutMode: WideString) [dispid $0000000C]; stdcall;  
procedure setNamedDest(none namedDest: WideString) [dispid $0000000D]; stdcall;  
procedure Print [dispid $0000000E]; stdcall;  
procedure printWithDialog [dispid $0000000F]; stdcall;  
procedure setZoom(none percent: Single) [dispid $00000010]; stdcall;  
procedure setZoomScroll(none percent, left, top: Single) [dispid $00000011]; stdcall;  
procedure setView(none viewMode: WideString) [dispid $00000012]; stdcall;  
procedure setViewScroll(none viewMode: WideString; none offset: Single) [dispid $00000013]; stdcall;  
procedure setViewRect(none left, top, width, height: Single) [dispid $00000014]; stdcall;  
procedure printPages(none from, to: Integer) [dispid $00000015]; stdcall;  
procedure printPagesFit(none from, to: Integer; none shrinkToFit: WordBool) [dispid $00000016];  
procedure printAll [dispid $00000017]; stdcall;  
procedure printAllFit(none shrinkToFit: WordBool) [dispid $00000018]; stdcall;  
procedure setShowScrollbars(none On: WordBool) [dispid $00000019]; stdcall;  
procedure AboutBox [dispid $FFFFFDD8]; stdcall;
```

But there is no documentation about them. You can download a free SDK (you need to register) but I believe

that you need the Acrobat package to fully use it.

<http://partners.adobe.com/asn/acrobat/download.jsp#fullinstall>

## Write a Window Log Event

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0209.html>

```
CONSTANT integer SUCCESS = 0
CONSTANT integer ERROR = 1
CONSTANT integer WARNING = 2
CONSTANT integer INFORMATION = 4
CONSTANT integer AUDIT_SUCCESS = 8
CONSTANT integer AUDIT_FAILURE = 16

OLEObject      wsh

wsh = CREATE OLEObject
wsh.ConnectToNewObject( "WScript.Shell" )
wsh.LogEvent( SUCCESS, "Application started" )
wsh.DisconnectObject()
DESTROY wsh
```

Note : Logevent can take a third argument which is a computer name where the event log is stored (the default is the local computer system). Applies to Windows NT/2000 only.

In Windows NT/2000, events are logged in the Windows NT Event Log. In Windows 9x/Me, events are logged in WSH.log (located in the Windows directory).

## Print a Word document

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0213.html>

```
oleObject ole_1
string ls_result
ole_1 = CREATE OLEObject
ole_1.ConnectToNewObject( "word.application" )
ole_1.documents.open("C:\realhowto.doc")
ole_1.selection.typetext(ls_result)
ole_1.activeworddocument.PrintOut(FALSE)
ole_1.activeworddocument.PrintOut()
ole_1.activeworddocument.Close(0)
```

```
ole_1.quit()
ole_1.disconnectObject()
DESTROY ole_1
```

## Remove HTML tags to keep only text (with IE)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0238.html>

```
OLEObject IE
OLEObject IEdoc
String ls_text

IE = CREATE OLEObject
IE.ConnectToNewObject("InternetExplorer.Application")

IE.left=200
IE.top=200
IE.height=400
IE.width=400
IE.menuBar=0
IE.toolbar=1
IE.statusBar=0
IE.navigate("http://www.rgagnon.com/donate.html")
IE.visible=0 // 1 to see IE

DO WHILE IE.busy
LOOP

ls_text = IE.Document.Body.innerText

MessageBox("HTML -> TEXT", ls_text)
// if needed, save the context of ls_text into File

IE.Quit()
```

## Get CPU or other hardware/software infos (with WMI)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0252.html>

Send Query to Windows using the WMI package.

To know more about your CPU

ref : [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_process](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_process)

```

class Win32_Processor : CIM_Processor
{
    uint16 AddressWidth;
    uint16 Architecture;
    uint16 Availability;
    string Caption;
    uint32 ConfigManagerErrorCode;
    boolean ConfigManagerUserConfig;
    uint16 CpuStatus;
    string CreationClassName;
    uint32 CurrentClockSpeed;
    uint16 CurrentVoltage;
    uint16 DataWidth;
    string Description;
    string DeviceID;
    boolean ErrorCleared;
    string ErrorDescription;
    uint32 ExtClock;
    uint16 Family;
    datetime InstallDate;
    uint32 L2CacheSize;
    uint32 L2CacheSpeed;
    uint32 LastErrorCode;
    uint16 Level;
    uint16 LoadPercentage;
    string Manufacturer;
    uint32 MaxClockSpeed;
    string Name;
    string OtherFamilyDescription;
    string PNPDeviceID;
    uint16 PowerManagementCapabilities[];
    boolean PowerManagementSupported;
    string ProcessorId;
    uint16 ProcessorType;
    uint16 Revision;
    string Role;
    string SocketDesignation;
    string Status;
    uint16 StatusInfo;
    string Stepping;
    string SystemCreationClassName;
    string SystemName;
    string UniqueId;
    uint16 UpgradeMethod;
    string Version;
    uint32 VoltageCaps;
};

[powerscript]

OLEObject ole_wsh
Any la_processor[]
string ls_message

```

```

ole_wsh = CREATE OLEObject
ole_wsh.ConnectToNewObject ("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function rtnProcessor()~r~n' &
+ 'DIM objProcessor(3)~r~n' &
+ 'strComputer = "."~r~n' &
+ 'Set objWMIService =' &
+ '    GetObject("winmgmts:\\" & strComputer & "\root\cimv2")~r~n' &
+ 'Set colItems =' &
+ '    objWMIService.ExecQuery("Select * from Win32_Processor")~r~n' &
+ 'For Each objItem in colItems~r~n' &
+ 'objProcessor(0) = objItem.ProcessorId~r~n' &
+ 'objProcessor(1) = objItem.MaxClockSpeed~r~n' &
+ 'objProcessor(2) = objItem.Name~r~n' &
+ 'objProcessor(3) = objItem.Description~r~n' &
+ 'Next~r~n' &
+ 'rtnProcessor = objProcessor~r~n' &
+ 'End Function')
la_processor[] = ole_wsh.Eval("rtnProcessor")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

ls_message = "Processor Id: " + string(la_processor[1]) + "~r~n" + &
+ "Maximum Clock Speed: " + string(la_processor[2]) + "~r~n" + &
+ "Name : " +string(la_processor[3]) + "~r~n" + &
+ "Description : " + string(la_processor[4])
MessageBox("Win32 Processor",ls_message)

```

To determine devices plugged into USB ports

```

[powerscript]
OLEObject ole_wsh
Any la_usb[]
string ls_message

ole_wsh = CREATE OLEObject

```

```

ole_wsh.ConnectToNewObject("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function rtnUSB()~r~n' &
+ 'USBdevice = "" ~r~n' &
+ 'strComputer = "."~r~n' &
+ 'Set objWMIService = ' &
+ 'GetObject("winmgmts:\\" & strComputer & "\root\cimv2")~r~n' &
+ 'Set colItems = ' &
+ 'objWMIService.ExecQuery("Select * from Win32_USBHub")~r~n' &
+ 'For Each objItem in colItems~r~n' &
+ 'USBdevice = USBdevice & " , " & objItem.Description~r~n' &
+ 'Next~r~n' &
+ 'rtnUSB = USBdevice~r~n' &
+ 'End Function')

ls_message = ole_wsh.Eval("rtnUSB")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

ls_message = "Device(s): " + ls_message
MessageBox("USB", ls_message)

```

Obtain the inventory all the software installed on a computer (can take awhile!)

```

OLEObject ole_wsh
Any la_usb[]
string ls_message

ole_wsh = CREATE OLEObject
ole_wsh.ConnectToNewObject("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function rtnSoftwares()~r~n' &
+ 'SoftwareList = "" ~r~n' + &
+ 'strComputer = "."~r~n' + &
+ 'Set objWMIService = ' + &
+ 'GetObject("winmgmts:{impersonationLevel=impersonate}!\\" _~r~n' &
+ '& strComputer & "\root\cimv2")~r~n' &

```

```

+ 'Set colItems = ' &
+ '  objWMIService.ExecQuery("Select * from Win32_Product")~r~n' &
+ 'For Each objItem in colItems~r~n' &
+ 'SoftwareList = SoftwareList & " , " & objItem.Name & + "(" & _~r~n' &
+ 'objItem.Version & ")"~r~n' &
+ 'Next~r~n' &
+ 'rtnSoftwares = SoftwareList~r~n' &
+ 'End Function')
ls_message = ole_wsh.Eval("rtnSoftwares")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

MessageBox("Software List",ls_message)

```

Returns Serial Number from BaseBoard

```

OLEObject ole_wsh
Any la_baseboard[]
string ls_message

ole_wsh = CREATE OLEObject
ole_wsh.ConnectToNewObject("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function rtnBaseBoard()~r~n' &

+ 'DIM objBaseBoard(2)~r~n' &
+ 'strComputer = "."~r~n' &
+ 'Set objWMIService =' &
+ '  GetObject("winmgmts:\\" & strComputer & "\root\cimv2")~r~n' &
+ 'Set colItems =' &
+ '  objWMIService.ExecQuery("SELECT Product, SerialNumber FROM Win32_BaseBoard")~r~n' &
+ 'For Each objItem in colItems~r~n' &
+ 'objBaseBoard(0) = objItem.Product~r~n' &
+ 'objBaseBoard(1) = objItem.SerialNumber~r~n' &
+ 'Next~r~n' &
+ 'rtnBaseBoard = objBaseBoard~r~n' &

```

```

+ 'End Function')
la_baseboard[] = ole_wsh.Eval("rtnBaseBoard")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

ls_message = "Product: " + string(la_baseboard[1]) + "~r~n" + &
+ "SerialNumber: " + string(la_baseboard[2]) + "~r~n"
MessageBox("Win32 BaseBoard",ls_message)

```

[more examples at MSDN](#)

Before using this technique, you may need to download and install the WMI ActiveX, here the [URL](#).

See this [related howto](#)

## Use MQSeries

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0203.html>

- You need the MQSeries ActiveX client installed on the workstation, you can get it freely from the [IBM Web site](#).
- An environment system variable (MQSERVER) must be defined.

The format is MQSERVER = NAME.OF.CONNECTION/PROTOCOL/NAME.OF.SERVER(PORT).

ex : MQSERVER = REMOTE.SVRCONN/TCP/MQSNT.SVR1.LOCAL(1414)

Powerscript exemple which write a message then read it back.

```

OLEObject MQS, QM, Q, ME, ML, PO, GO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject      // MQSeries Session
QM = CREATE OLEObject      // QueueManager
Q = CREATE OLEObject       // Queue
MW = CREATE OLEObject      // Message Written
MR = CREATE OLEObject      // Message Read
PO = CREATE OLEObject      // Put options
GO = CREATE OLEObject      // Get options

MQS.ConnectToNewObject ("MQAX200.MqSession") // see note 1

ls_mqmgr = "MQS.CORP.DEPT.DEV"           // see note 2
QM = MQS.AccessQueueManager(ls_mqmgr)
Q = QM.AccessQueue ("CORP.DEPT.SYST.STAGE.QNP1",1 + 16) // see note 3
// WRITE
MW = MQS.AccessMessage

```

```

MW.MessageData = "HELLO WORLD " + string(cpu())
PO = MQS.AccessPutMessageOptions()
Q.Put( MW, PO)
// READ
MR = MQS.AccessMessage()
MR.MessageIdHex = MR.MessageIdHex // see note 4
GO = MQS.AccessGetMessageOptions()
Q.Get( MR, GO)
ls_s = MR.ReadString(MR.MessageLength)
MessageBox("MQSeries", ls_s)

Q.close()
QM.disconnect()

```

**NOTE:**

1. Connection to the ActiveX
2. The Queue Manager
3. The Queue Name , see the constant defintion below, here the Queue is INPUT and OUTPUT
4. Just to make sure that we are reading the good message.

The possible constants are :

## Message TYPE

```

Global Const MQMT_SYSTEM_FIRST = 1
Global Const MQMT_REQUEST = 1
Global Const MQMT_REPLY = 2
Global Const MQMT_DATAGRAM = 8
Global Const MQMT_REPORT = 4
Global Const MQMT_SYSTEM_LAST = 65535
Global Const MQMT_APPL_FIRST = 65536
Global Const MQMT_APPL_LAST = 999999999

```

## Open Options

```

Global Const MQOO_INPUT_AS_Q_DEF = 1
Global Const MQOO_INPUT_SHARED = 2
Global Const MQOO_INPUT_EXCLUSIVE = 4
Global Const MQOO_BROWSE = 8
Global Const MQOO_OUTPUT = 16
Global Const MQOO_INQUIRE = 32
Global Const MQOO_SET = 64
Global Const MQOO_SAVE_ALL_CONTEXT = 128
Global Const MQOO_PASS_IDENTITY_CONTEXT = 256
Global Const MQOO_PASS_ALL_CONTEXT = 512
Global Const MQOO_SET_IDENTITY_CONTEXT = 1024
Global Const MQOO_SET_ALL_CONTEXT = 2048
Global Const MQOO_ALTERNATE_USER_AUTHORITY = 4096
Global Const MQOO_FAIL_IF_QUIESCING = 8192

```

## Put Message Options

```

Global Const MQPMO_SYNCPOINT = 2
Global Const MQPMO_NO_SYNCPOINT = 4
Global Const MQPMO_NEW_MSG_ID = 64
Global Const MQPMO_NEW_CORREL_ID = 128
Global Const MQPMO_LOGICAL_ORDER = 32768
Global Const MQPMO_NO_CONTEXT = 16384
Global Const MQPMO_DEFAULT_CONTEXT = 32
Global Const MQPMO_PASS_IDENTITY_CONTEXT = 256
Global Const MQPMO_PASS_ALL_CONTEXT = 512
Global Const MQPMO_SET_IDENTITY_CONTEXT = 1024
Global Const MQPMO_SET_ALL_CONTEXT = 2048
Global Const MQPMO_ALTERNATE_USER_AUTHORITY = 4096
Global Const MQPMO_FAIL_IF QUIESCING = 8192
Global Const MQPMO_NONE = 0

```

#### Get Message Options

```

Global Const MQGMO_WAIT = 1
Global Const MQGMO_NO_WAIT = 0
Global Const MQGMO_SYNCPOINT = 2
Global Const MQGMO_SYNCPOINT_IF_PERSISTENT = 4096
Global Const MQGMO_NO_SYNCPOINT = 4
Global Const MQGMO_MARK_SKIP_BACKOUT = 128
Global Const MQGMO_BROWSE_FIRST = 16
Global Const MQGMO_BROWSE_NEXT = 32
Global Const MQGMO_BROWSE_MSG_UNDER_CURSOR = 2048
Global Const MQGMO_MSG_UNDER_CURSOR = 256
Global Const MQGMO_LOCK = 512
Global Const MQGMO_UNLOCK = 1024
Global Const MQGMO_ACCEPT_TRUNCATED_MSG = 64
Global Const MQGMO_SET_SIGNAL = 8
Global Const MQGMO_FAIL_IF QUIESCING = 8192
Global Const MQGMO_CONVERT = 16384
Global Const MQGMO_LOGICAL_ORDER = 32768
Global Const MQGMO_COMPLETE_MSG = 65536
Global Const MQGMO_ALL_MSGS_AVAILABLE = 131072
Global Const MQGMO_ALL_SEGMENTS_AVAILABLE = 262144
Global Const MQGMO_NONE = 0

```

Powerscript example reading from a Queue but with a check if the Queue is empty.

```

OLEObject MQS, QM, Q, ML, GO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject
QM = CREATE OLEObject
Q = CREATE OLEObject
MR = CREATE OLEObject
GO = CREATE OLEObject

MQS.ConnectToNewObject ("MQAX200.MqSession")

ls_mqmgr = "MQS.CORP.DEPT.DEV"
QM = MQS.AccessQueueManager(ls_mqmgr)

```

```

Q = QM.AccessQueue("CORP.DEPT.SYST.STAGE.QNP1", 1 + 32) // INPUT & INQUIRE

IF Q.CurrentDepth = 0 THEN
    MessageBox("", "The Queue is empty")
ELSE
    ls_s = string(Q.CurrentDepth)
    MessageBox("", "The Queue has " + ls_s + " element(s)")

    ML = MQS.AccessMessage()
    MR.MessageType = 77777 // our known message type
    GO = MQS.AccessGetMessageOptions()
    GO.Options = GO.Options + 4 // GOO_NO_SYNCPOINT
    Q.Get( MR, GO )

    ls_s = MR.ReadString(ML.MessageLength)
    MessageBox("MQSeries", ls_s)
END IF

Q.close()
QM.disconnect()

```

The next example is two applications talking each other.

The first writes a message into a Queue with a known type. The second application reads at regular interval the Queue for available messages and prints them into scrolling listbox.

First application :

```

[button script]
OLEObject MQS, QM, Q, MW, PO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject
QM = CREATE OLEObject
Q = CREATE OLEObject
MW = CREATE OLEObject
PO = CREATE OLEObject

MQS.ConnectToNewObject("MQAX200.MqSession")

ls_mqmgr = "MQS.CORP.DEPT.DEV"
QM = MQS.AccessQueueManager(ls_mqmgr)
Q = QM.AccessQueue("CORP.DEPT.SYST.STAGE.QNP1", 16) // OUTPUT

MW = MQS.AccessMessage
MW.MessageData = "HELLO WORLD " + string(cpu())
MW.MessageType = 77778
MW.Format = "MQSTR"
PO = MQS.AccessPutMessageOptions()
Q.Put( MW, PO)
Q.close()

```

```
QM.disconnect()
```

## Application 2

```
[window instance variable]
OLEObject iMQS, iQM, iQ, iMR, iGO
String is_mqmgr, is_s

[open window event]
iMQS = CREATE OLEObject
iQM = CREATE OLEObject
iQ = CREATE OLEObject
iMR = CREATE OLEObject
iGO = CREATE OLEObject

iMQS.ConnectToNewObject ("MQAX200.MqSession")

is_mqmgr = "MQS.CORP.DEPT.DEV"
iQM = iMQS.AccessQueueManager(is_mqmgr)
timer(1)

[close window event]
// QUEUE MANAGER DISCONNECT
// iQM.disconnect()

[timer window event]
// write into a listbox (w=1563 h=572 not sorted vscrollbar)
integer cpt = 0

iQ = iQM.AccessQueue ("CORP.DEPT.SYST.STAGE.QNP1",1 + 32) // INPUT,INQUIRE

IF iQ.CurrentDepth = 0 THEN
    cpt = lb_1.additem("The Queue is empty")
ELSE
    iMR = iMQS.AccessMessage()
    iMR.MessageType = 77778
    iGO = iMQS.AccessGetMessageOptions()
    iGO.Options = iGO.OPTIONS + 4 // GO_NO_SYNCPOINT
    iQ.Get( iMR, iGO )

    is_s = iMR.ReadString(iML.MessageLength)
    cpt = lb_1.additem(is_s)
END IF

IF cpt > 8 THEN
    lb_1.SetTop(cpt - 8 + 1)
END IF

// QUEUE close
// Q.close()
```

# Terminate a Windows application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0266.html>

## Using VBScript

This will immediately close any running instance of Notepad.

```
OleObject wsh
integer li_rc

wsh = CREATE OleObject
wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"
wsh.AddCode('function terminatenotepad() ~n ' + &
'strComputer = "." ~n ' + &
'Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2") ~n ' + &
'Set colItems = objWMIService.ExecQuery("Select * from Win32_Process where name = ~'notepad.
'For Each objItem in colItems ~n ' + &
'    objItem.Terminate ~n ' + &
'Next ~n ' + &
'end function ~n ')
wsh.executestatement('terminatenotepad()')
wsh.DisconnectObject()
DESTROY wsh
```

Replace notepad.exe with the executable name that you want to close.

## Using WinAPI with the class name

This will close a running Notepad but will ask to save any unsaved document. We are using the WinAPI to retrieve an handle and then we send the event WM\_CLOSE to the object associated with the handle.

```
[local external function declaration]
// PB10
FUNCTION ulong FindWindow(ref string classname, ref string windowname) &
LIBRARY "user32.dll" &
ALIAS FOR "FindWindowA;ansi"

// or PB10 alternate declaration
// FUNCTION ulong FindWindow (ref string classname, ref string windowname) &
// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowW"

// or PB9
// FUNCTION ulong FindWindow(ref string classname, ref string windowname) &
```

```

// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowA"

[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_class = "Notepad"
SetNull(ls_name)

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10 == 16
    send(hwnd, 16, 0, 0)
END IF

```

You need a **special tool** to detect the class name. A well known tool to detect the class name is SPY++, you may already have it (since it was included with PB Enterprise). Or you can download a free one, WinID at <http://www.softpedia.com/get/Programming/Debuggers-Decompilers-Dissassemblers/WinID.shtml>

To close an application builded with Powerbuilder, you need a class name, it's difficult because the class name is not always the same with new PB release. With PB10, it's FNWND3100 so for PB11 it maybe FNWND3110 ...

PB9	ls_class = "FNWND390"
PB10	ls_class = "FNWND3100"
PB11	ls_class = "FNWND3110" // ??

You may need to use the next technique if you want to close another Powerbuilder application and not close yourself at the same time !

### Using WinAPI with a window title

Another possibility is to use the window title with the FindWindow API to search for the handle. This is ok if your title is static and known in advance.

```

[local external function declaration]
// PB10
FUNCTION ulong FindWindow(ref string classname, ref string windowname) LIBRARY "user32.dll"
ALIAS FOR "FindWindowA;ansi"

// or PB10 alternate declaration
// FUNCTION ulong FindWindow (ref string classname, ref string windowname) &
// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowW"

// or PB9

```

```
// FUNCTION ulong FindWindow(ref string classname, ref string windowname)  &
// LIBRARY "user32.dll"  &
// ALIAS FOR "FindWindowA"

[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_name = "Untitled - Notepad"
SetNull(ls_class)

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10
    send(hwnd, 16, 0, 0)
END IF
```

Note : It's possible to search the handle with the class name and the window title.

```
[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_name = "Untitled - Notepad"
ls_class= "Notepad"

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10
    send(hwnd, 16, 0, 0)
END IF
```

## Get the current user email using Active Directory

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-get-the-current-user-email-using-active-directory.html>

```
OLEObject ole_wsh
string ls_email

ole_wsh = CREATE OLEObject
ole_wsh.ConnectToNewObject("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function getEmail()~r~n' &
```

```

+ 'Dim objADSysInfo~r~n'  &
+ 'Set objADSysInfo = CreateObject("ADSystemInfo")~r~n'  &
+ 'Dim objUser~r~n'  &
+ 'Set objUser = GetObject("LDAP://" & objADSysInfo.UserName)~r~n'  &
+ 'getEmail = objUser.Mail~r~n'  &
+ 'End Function')

ls_email = ole_wsh.Eval("getEmail")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

MessageBox("Your email",ls_email)

```

## Get a value from a JSON String

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/powerbuilder-get-a-value-from-JSON-String.html>

Parsing a JSON string to retrieve a value is not something easy in Powerscript. One way to achieve that is to embed a JScript code in Powerscript using the COM object *MSScriptControl.ScriptControl* and let the JScript engine do the work. A real JSON parser is not available with the script engine is used this way, but the eval() function can be used for simple JSON string (see the **important note** below).

```

OleObject lole_ScriptControl
String ls_json
String ls_value

lole_ScriptControl = CREATE OleObject
lole_ScriptControl.ConnectToNewObject( "MSScriptControl.ScriptControl" )
lole_ScriptControl.Language = "JScript"
lole_ScriptControl.AddCode("function getValue(s,key) {eval(~"jsonobj=~" + s); return eval(~"j

//      key      value
ls_json = "{ test1 : 'value1' , test2 : 'value2' }"
TRY
    ls_value =  lole_ScriptControl.Eval ("getValue(~" +ls_json + "~", ~"test1~");")
    MessageBox("", ls_value)
    ls_value =  lole_ScriptControl.Eval ("getValue(~" +ls_json + "~", ~"test2~");")
    MessageBox("", ls_value)
CATCH ( Throwable e )
    MessageBox("Err", e.GetMessage())
END TRY

```

Since the JSON string has single quote to represent strings, I use double quotes to pass the JSON string to the JScript function.

### IMPORTANT NOTE

You have to make sure that the JSON string is valid and from a **trusted source** because this technique (using eval() and not a real JSON parser) is not safe and can be used to inject and execute malicious code.

See also this HowTo : [Get a value from a REST service](#)

## Get a value from a REST service

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/powerbuilder-get-a-value-from-REST-Service.html>

From this [HowTo](#), we saw how to get a value from a JSON string. Habitually, we get a JSON string from a Web service exposed as REST service. It maybe difficult for Powerbuilder to call this kind of service because there is no way to specify HTTP header with regular Powerscript.

Fortunately, we can use two Windows COM objects, *Msxml2.DOMDocument* and *MSXML2.ServerXMLHTTP*, to do that.

For demonstration, we call a REST service that return the current date and time. This service is hosted at <http://www.jsontest.com/>. See the **important note** below.

```
OleObject lole_Send
OleObject lole_SrvHTTP
OleObject lole_ScriptControl

String ls_status
String ls_response
String ls_value

lole_Send = CREATE OleObject
lole_SrvHTTP = CREATE OleObject
lole_Send.connectToNewObject("Msxml2.DOMDocument.6.0")
lole_SrvHTTP.connectToNewObject("MSXML2.ServerXMLHTTP.6.0")
lole_SrvHTTP.Open("GET", "http://date.jsontest.com", FALSE)
lole_SrvHTTP.SetRequestHeader( "Content-Type", "application/json")
lole_SrvHTTP.Send(lole_Send)
ls_status = string(lole_SrvHTTP.Status)
ls_response = string(lole_SrvHTTP.ResponseText)

MessageBox("REST HTTP response", ls_status) // 200 is OK!
MessageBox("JSON response", ls_response)

lole_ScriptControl = CREATE OleObject
lole_ScriptControl.ConnectToNewObject( "MSScriptControl.ScriptControl" )
lole_ScriptControl.Language = "JScript"
lole_ScriptControl.AddCode("function getValue(s,key) {eval(~"jsonobj=~" + s); return eval(~"j

TRY
// remove (replace with "") all the carriage return to clean up the JSON string
int position, i
```

```

FOR i = 1 to len(ls_response)
    position = pos(ls_response, "~n")
    IF position > 0 THEN
        ls_response= Replace(ls_response, position, 1, "")
    END IF
NEXT
MessageBox("JSON response cleanup", ls_response)
// typical response
// { "time": "04:21:52 PM", "milliseconds_since_epoch": 1436113312190, "date": "07-05-2015" }

ls_value = lole_ScriptControl.Eval ("getValue(' " + ls_response + " ', 'date') ;")
MessageBox("date value", ls_value)
CATCH ( Throwable e )
    MessageBox("Err", e.GetMessage())
END TRY

```

Since the JSON string has quotes to represent strings, I use single quotes to pass the JSON string to the JScript function.

### **IMPORTANT NOTE**

You have to make sure that the JSON string is valid and from a **trusted source** because this technique (using eval() and not a real JSON parser) is not safe and can be used to inject and execute malicious code.

See also this HowTo : [Get a value from a JSON string](#).

## **Start the default browser**

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0009.html>

There are many ways to start a browser from PB.

You can use the file association (see [previous How-to](#)).

Specify the browser executable path :

```

string url,fix
SetNull(url)
theBrowser = '"C:\Program Files\Internet Explorer\iexplore.exe" -nohome'
theUrl = sle_1.text
IF NOT IsNull(theUrl) THEN
    Run(theBrowser + " " + theUrl)
END IF

```

With PB v6, the new Internet object provides some facilities to accomplish this.

```
string url
```

```

inet iinet_base

url = sle_1.text
GetContextService("Internet", iinet_base)
iinet_base.HyperlinkToUrl(url)

```

See this [How-to](#) for Internet Explorer specific snippets.

## Upload a file using FTP

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0013.html>

```

[DOS Batch file upload.bat]
@rem This file will attempt to upload 1 files to the ftp server
@c:
@cd\temp
@echo open ftpserveripaddress >> t
@echo userid >> t
@echo password >> t
@echo binary >> t
@echo cd uploaddirectoryname >> t
@echo mput filenametobeuploaded >> t
@echo bye >> t
@ftp -i -s:t
@del t

[PB external function declaration]
FUNCTION long ShellExecute (uint ihwnd, string lpszOp, string
    lpszFile, string lpszParams, string lpszDir, int wShowCmd )
LIBRARY "Shell32.dll" ALIAS FOR "ShellExecuteA"

[PB ftp upload example]
string ls_parm, ls_dir
uint ll_ret
ls_parm = ""
ls_dir = "c:\path\"

ll_ret = ShellExecute &

(handle(this), "open", "upload.bat", ls_parm, ls_dir, 0)
IF ll_ret <32 THEN
    MessageBox( "cannot load" , ll_ret )
END IF

```

## Start a dial-up connection

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0021.html>

```
string command  
  
command='rundll rnaui.dll,RnaDial YourConnection' // case sensitive  
Run(command)
```

## Start/Run services

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0148.html>

```
run( "NET START ServiceName" )  
run( "NET STOP ServiceName" )
```

## Display default email window to type and send email

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0160.html>

```
run("rundll32 url.dll,FileProtocolHandler " + &  
    "mailto:real@rgagnon.com&subject=HelloWorld")
```

## Use Internet Explorer

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0202.html>

- Method #1 - Using HyperlinkToURL function

Needs PB6.5 or better. Starts the default browser as defined in your installation.

```
string ls_url
```

```
inet iinet_base  
  
ls_url = "http://www.server.com/myWebApp?file=error.log"  
GetContextService("Internet", iinet_base)  
iinet_base.HyperlinkToURL(ls_url)
```

- Method #2 - Using OLEObject InternetExplorer.Application + Run

```
OLEObject IE  
string ls_ie  
string ls_url  
  
IE = CREATE OLEObject  
IE.ConnectToNewObject("InternetExplorer.Application")  
  
ls_ie = IE.Fullname()  
  
ls_url = "http://www.server.com/myWebApp?file=error.log"  
Run (ls_ie + " " + ls_url)
```

- Method #3 - Using OLEObject InternetExplorer.Application

Provides complete control IE behaviour.

```
OLEObject IE  
  
IE = CREATE OLEObject  
IE.ConnectToNewObject("InternetExplorer.Application")  
  
IE.left=200  
IE.top=200  
IE.height=400  
IE.width=400  
IE.menuBar=0  
IE.toolbar=1  
IE.statusBar=0  
IE.navigate("http://www.server.com/myWebApp?file=error.log")  
IE.visible=1  
  
// Local external function decalration  
// FUNCTION boolean SetForegroundWindow( long hWnd ) LIBRARY "USER32"  
SetForegroundWindow( IE.HWND )
```

- Method 4 - Using OLEObject InternetExplorer.Application

Provides complete control IE behaviour. You simulate a FORM submit using the POST method.  
OLEObject IE

```
IE = CREATE OLEObject  
IE.ConnectToNewObject("InternetExplorer.Application")  
  
IE.left=200  
IE.top=200  
IE.height=400
```

```

IE.width=400
IE.menuBar=0
IE.toolbar=1
IE.statusBar=0
IE.goHome
IE.visible=1

DO WHILE IE.busy
LOOP

IE.document.Open
IE.document.WriteLine("<HTML><HEAD>")
IE.document.WriteLine("<SCRIPT>")
IE.document.WriteLine("function go() {")
IE.document.WriteLine("  document.myform.submit()")
IE.document.WriteLine("}")
IE.document.WriteLine("</SCRIPT>")
IE.document.WriteLine("</HEAD>")
IE.document.WriteLine("<BODY onload='go()'>")
IE.document.WriteLine("<FORM name='myform' >")
IE.document.WriteLine("METHOD='GET' ACTION='http://server.com/myApp'")
IE.document.WriteLine("<INPUT NAME='file' TYPE=HIDDEN VALUE='error.log'>")
IE.document.WriteLine("</FORM>")
IE.document.WriteLine("</BODY>")
IE.document.WriteLine("</HTML>")
IE.document.Close

// Local external function declaration
// FUNCTION boolean SetForegroundWindow( long hWnd ) LIBRARY "USER32"
SetForegroundWindow( IE.HWND )



// IE.Quit()
// IE.DisconnectObject()
// DESTROY iBrowser

```

---

Some tips from rkern (**thanks** to him!).

Some handy properties:

```

IE.top = <integer>
IE.left = <integer>
IE.width = <integer>
IE.height = <integer>
IE.visible = <integer> (1 - Visible/ 0 - Not Visible)
IE.menuBar = <integer> (1 - Visible/ 0 - Not Visible)
IE.toolBar = <integer> (1 - Visible/ 0 - Not Visible)
IE.statusBar = <integer> (1 - Visible/ 0 - Not Visible)

```

Go To a URL:

```

string ls_OldURL
string ls_NewURL = 'www.yahoo.com'
// Go to Document
ls_OldURL = IE.LocationURL
IE.Navigate (ls_NewURL)
// Wait for Document to start loading
DO WHILE ls_OldURL = IE.LocationURL
    Yield ()
LOOP
// Wait for Document to finish Loading
DO WHILE IE.Busy
    Yield ()
LOOP

```

Notes:

- The Busy attribute does not always get set immediately so wait for URL to change before checking the Busy attribute
- This method works as long as the OldURL and NewURL are not the same

**Click on a Link with a specific display text:**

```

integer li_link, li_link_ctr
string ls_text, ls_OldURL, ls_search = 'Download Spreadsheet'
// Set Match string
ls_search = Upper(ls_search)
// Find Link with specified Text
li_link_ctr = IE.Document.Links.Length
FOR li_link = 0 TO li_link_ctr - 1
    ls_Text = Upper(IE.Document.Links[li_link].InnerText)
    // Check for Match
    IF ls_Text = as_Search THEN
        // Click the Link
        ls_OldURL = IE.LocationURL
        IE.Document.Links[li_link].Click()
        // Wait for Document to start loading
        DO WHILE ls_OldURL = IE.LocationURL
            Yield ()
        LOOP
        // Wait for Document to finish loading
        DO WHILE IE.Busy
            Yield ()
        LOOP
        EXIT
    END IF
NEXT

```

Notes:

- The Busy attribute does not always get set immediately so wait for URL to change before checking the Busy attribute
- This method works as long as the OldURL and NewURL are not the same - Unlike PB, arrays start at index = 0 to length - 1 - Use attribute HRef if you want to match on a URL

**Click on a Button with a specific display text:**

```

integer li_form, li_form_ctr
integer li_elem, li_elem_ctr

```

```

boolean lb_stop
string ls_text, ls_OldURL, ls_search = 'Search'
any la_name, la_type, la_value
// Set Match string
ls_search = Upper(ls_search)
// Search thru Forms
li_form_ctr = IE.Document.Forms.Length
FOR i = 0 TO li_form_ctr - 1
    // Search thru Elements
    li_elem_ctr = IE.Document.Forms[i].Elements.All.Length
    FOR j = 0 TO li_elem_ctr - 1
        la_Name =
            IE.Document.Forms[i].Elements.All[j].GetAttribute("name")
        la_Type =
            IE.Document.Forms[i].Elements.All[j].GetAttribute("type")
        la_Value =
            IE.Document.Forms[i].Elements.All[j].GetAttribute("value")
        CHOOSE CASE Upper(String(la_Type))
        CASE 'SUBMIT', 'RESET', 'BUTTON'
            ls_Text = Upper(String(la_Value))
            // Check for Match
            IF ls_Text = ls_Search THEN
                // Click the Button
                ls_OldURL = IE.LocationURL
                IE.Document.Forms[i].Elements.All[j].Click()
                // Wait for Document to start loading
                DO WHILE ls_OldURL = IE.LocationURL
                    Yield ()
                LOOP
                // Wait for Document to finish loading
                DO WHILE IE.Busy
                    Yield ()
                LOOP
                lb_Stop = TRUE
                EXIT
            END IF
        END CHOOSE
    NEXT
    IF lb_Stop THEN
        EXIT
    END IF
NEXT

```

## Notes:

- The Busy attribute does not always get set immediately so wait for URL to change before checking the Busy attribute
- This method works as long as the OldURL and NewURL are not the same
- Unlike PB, arrays start at index = 0 to length - 1
- Use attribute Name if you want to match on an internal HTML Name
- The Returned Values of the GetAttribute method must be ANY variables , even though they return STRINGS

Fill in a Input Field that has a specific Name:

```

integer li_form, li_form_ctr
integer li_elem, li_elem_ctr
boolean lb_Stop

```

```

string ls_text, ls_search = 'Company'
any la_name, la_type, la_value
// Set Match string
ls_search = Upper(ls_search)
// Search thru Forms
li_form_ctr = IE.Document.Forms.Length
FOR i = 0 TO li_form_ctr - 1
    // Search thru Elements
    li_elem_ctr = IE.Document.Forms[li_form].Elements.All.Length
    FOR j = 0 TO li_elem_ctr - 1
        la_Name =
            IE.Document.Forms[i].Elements.All[j].getAttribute("name")
        la_Type =
            IE.Document.Forms[i].Elements.All[j].getAttribute("type")
        la_Value =
            IE.Document.Forms[i].Elements.All[j].getAttribute("value")
        // ls_Text = IE.Document.Forms[i].Elements.All[j].InnerText

        CHOOSE CASE Upper(String(la_Type))
        CASE 'TEXT', 'TEXTAREA', 'PASSWORD'
            ls_Text = Upper(String(la_Name))
            // Check for Match
            IF ls_Text = ls_Search THEN
                // Set the value
                IE.Document.Forms[i].Elements.All[j].Value = ls_Value
                lb_Stop = TRUE
                Exit
            END IF
        END CHOOSE
    NEXT
    IF lb_Stop THEN
        Exit
    END IF
NEXT

```

#### Notes:

- Unlike PB, arrays start at index = 0 to length - 1
- Use attribute InnerText if you want to match on a perceived label on the field. Check each element's InnerText until a match is found and then fill in the next input field you come to (usually works).
- The Returned Values of the GetAttribute method must be ANY variables , even though they return STRINGS

## Use MQSeries

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/pb-0203.html>

- You need the MQSeries ActiveX client installed on the workstation, you can get it freely from the [IBM Web site](#).
- An environment system variable (MQSERVER) must be defined.

The format is MQSERVER = NAME.OF.CONNECTION/PROTOCOL/NAME.OF.SERVER(PORT).

ex : MQSERVER = REMOTE.SVRCCONN/TCP/MQSNT.SVR1.LOCAL(1414)

Powerscript exemple which write a message then read it back.

```

OLEObject MQS, QM, Q, ME, ML, PO, GO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject      // MQSeries Session
QM = CREATE OLEObject       // QueueManager
Q = CREATE OLEObject        // Queue
MW = CREATE OLEObject       // Message Written
MR = CREATE OLEObject       // Message Read
PO = CREATE OLEObject       // Put options
GO = CREATE OLEObject       // Get options

MQS.ConnectToNewObject ("MQAX200.MqSession") // see note 1

ls_mqmgr = "MQS.CORP.DEPT.DEV"           // see note 2
QM = MQS.AccessQueueManager(ls_mqmgr)
Q = QM.AccessQueue ("CORP.DEPT.SYST.STAGE.QNP1",1 + 16) // see note 3
// WRITE
MW = MQS.AccessMessage
MW.MessageData = "HELLO WORLD " + string(cpu())
PO = MQS.AccessPutMessageOptions()
Q.Put( MW, PO)
// READ
MR = MQS.AccessMessage()
MR.MessageIdHex = MR.MessageIdHex      // see note 4
GO = MQS.AccessGetMessageOptions()
Q.Get( MR, GO)
ls_s = MR.ReadString(MR.MessageLength)
MessageBox ("MQSeries", ls_s)

Q.close()
QM.disconnect()

```

#### NOTE:

1. Connection to the ActiveX
2. The Queue Manager
3. The Queue Name , see the constant defintion below, here the Queue is INPUT and OUTPUT
4. Just to make sure that we are reading the good message.

The possible constants are :

```

Message TYPE

Global Const MQMT_SYSTEM_FIRST = 1
Global Const MQMT_REQUEST = 1
Global Const MQMT_REPLY = 2
Global Const MQMT_DATAGRAM = 8
Global Const MQMT_REPORT = 4
Global Const MQMT_SYSTEM_LAST = 65535

```

```
Global Const MQMT_APPL_FIRST = 65536
Global Const MQMT_APPL_LAST = 999999999
```

#### Open Options

```
Global Const MQOO_INPUT_AS_Q_DEF = 1
Global Const MQOO_INPUT_SHARED = 2
Global Const MQOO_INPUT_EXCLUSIVE = 4
Global Const MQOO_BROWSE = 8
Global Const MQOO_OUTPUT = 16
Global Const MQOO_INQUIRE = 32
Global Const MQOO_SET = 64
Global Const MQOO_SAVE_ALL_CONTEXT = 128
Global Const MQOO_PASS_IDENTITY_CONTEXT = 256
Global Const MQOO_PASS_ALL_CONTEXT = 512
Global Const MQOO_SET_IDENTITY_CONTEXT = 1024
Global Const MQOO_SET_ALL_CONTEXT = 2048
Global Const MQOO_ALTERNATE_USER_AUTHORITY = 4096
Global Const MQOO_FAIL_IF_QUIESCING = 8192
```

#### Put Message Options

```
Global Const MQPMO_SYNCPOINT = 2
Global Const MQPMO_NO_SYNCPOINT = 4
Global Const MQPMO_NEW_MSG_ID = 64
Global Const MQPMO_NEW_CORREL_ID = 128
Global Const MQPMO_LOGICAL_ORDER = 32768
Global Const MQPMO_NO_CONTEXT = 16384
Global Const MQPMO_DEFAULT_CONTEXT = 32
Global Const MQPMO_PASS_IDENTITY_CONTEXT = 256
Global Const MQPMO_PASS_ALL_CONTEXT = 512
Global Const MQPMO_SET_IDENTITY_CONTEXT = 1024
Global Const MQPMO_SET_ALL_CONTEXT = 2048
Global Const MQPMO_ALTERNATE_USER_AUTHORITY = 4096
Global Const MQPMO_FAIL_IF_QUIESCING = 8192
Global Const MQPMO_NONE = 0
```

#### Get Message Options

```
Global Const MQGMO_WAIT = 1
Global Const MQGMO_NO_WAIT = 0
Global Const MQGMO_SYNCPOINT = 2
Global Const MQGMO_SYNCPOINT_IF_PERSISTENT = 4096
Global Const MQGMO_NO_SYNCPOINT = 4
Global Const MQGMO_MARK_SKIP_BACKOUT = 128
Global Const MQGMO_BROWSE_FIRST = 16
Global Const MQGMO_BROWSE_NEXT = 32
Global Const MQGMO_BROWSE_MSG_UNDER_CURSOR = 2048
Global Const MQGMO_MSG_UNDER_CURSOR = 256
Global Const MQGMO_LOCK = 512
Global Const MQGMO_UNLOCK = 1024
Global Const MQGMO_ACCEPT_TRUNCATED_MSG = 64
Global Const MQGMO_SET_SIGNAL = 8
```

```

Global Const MQGMO_FAIL_IF_QUIESCING = 8192
Global Const MQGMO_CONVERT = 16384
Global Const MQGMO_LOGICAL_ORDER = 32768
Global Const MQGMO_COMPLETE_MSG = 65536
Global Const MQGMO_ALL_MSGS_AVAILABLE = 131072
Global Const MQGMO_ALL_SEGMENTS_AVAILABLE = 262144
Global Const MQGMO_NONE = 0

```

Powerscript example reading from a Queue but with a check if the Queue is empty.

```

OLEObject MQS, QM, Q, ML, GO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject
QM = CREATE OLEObject
Q = CREATE OLEObject
MR = CREATE OLEObject
GO = CREATE OLEObject

MQS.ConnectToNewObject ("MQAX200.MqSession")

ls_mqmgr = "MQS.CORP.DEPT.DEV"
QM = MQS.AccessQueueManager(ls_mqmgr)
Q = QM.AccessQueue ("CORP.DEPT.SYST.STAGE.QNP1", 1 + 32) // INPUT & INQUIRE

IF Q.CurrentDepth = 0 THEN
    MessageBox("", "The Queue is empty")
ELSE
    ls_s = string(Q.CurrentDepth)
    MessageBox("", "The Queue has " + ls_s + " element(s)")

    ML = MQS.AccessMessage()
    MR.MessageType = 7777 // our known message type
    GO = MQS.AccessGetMessageOptions()
    GO.Options = GO.OPTIONS + 4 // GO_NO_SYNCPOINT
    Q.Get( MR, GO )

    ls_s = MR.ReadString(ML.MessageLength)
    MessageBox("MQSeries", ls_s)
END IF

Q.close()
QM.disconnect()

```

The next example is two applications talking each other.

The first writes a message into a Queue with a known type. The second application reads at regular interval the Queue for available messages and prints them into scrolling listbox.

First application :

```
[button script]
OLEObject MQS, QM, Q, MW, PO
String ls_mqmgr, ls_s

MQS = CREATE OLEObject
QM = CREATE OLEObject
Q = CREATE OLEObject
MW = CREATE OLEObject
PO = CREATE OLEObject

MQS.ConnectToNewObject ("MQAX200.MqSession")

ls_mqmgr = "MQS.CORP.DEPT.DEV"
QM = MQS.AccessQueueManager(ls_mqmgr)
Q = QM.AccessQueue ("CORP.DEPT.SYST.STAGE.QNP1",16) // OUTPUT

MW = MQS.AccessMessage
MW.MessageData = "HELLO WORLD " + string(cpu())
MW.MessageType = 77778
MW.Format = "MQSTR"
PO = MQS.AccessPutMessageOptions()
Q.Put( MW, PO)
Q.close()
QM.disconnect()
```

## Application 2

```
[window instance variable]
OLEObject iMQS, iQM, iQ, iMR, iGO
String is_mqmgr, is_s

[open window event]
iMQS = CREATE OLEObject
iQM = CREATE OLEObject
iQ = CREATE OLEObject
iMR = CREATE OLEObject
iGO = CREATE OLEObject

iMQS.ConnectToNewObject ("MQAX200.MqSession")

is_mqmgr = "MQS.CORP.DEPT.DEV"
iQM = iMQS.AccessQueueManager(is_mqmgr)
timer(1)

[close window event]
// QUEUE MANAGER DISCONNECT
// iQM.disconnect()

[timer window event]
// write into a listbox (w=1563 h=572 not sorted vscrollbar)
integer cpt = 0
```

```

iQ = iQM.AccessQueue("CORP.DEPT.SYST.STAGE.QNP1",1 + 32) // INPUT, INQUIRE

IF iQ.CurrentDepth = 0 THEN
    cpt = lb_1.additem("The Queue is empty")
ELSE
    iMR = iMQS.AccessMessage()
    iMR.MessageType = 77778
    iGO = iMQS.AccessGetMessageOptions()
    iGO.Options = iGO.Options + 4 // GOO_NO_SYNCPOINT
    iQ.Get( iMR, iGO )

    is_s = iMR.ReadString(iML.MessageLength)
    cpt = lb_1.additem(is_s)
END IF

IF cpt > 8 THEN
    lb_1.SetTop(cpt - 8 + 1)
END IF

// QUEUE close
// Q.close()

```

## Use RunDll32 utility

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0204.html>

RunDll32 executable can be used to start various Windows utilities like the Control Panel. Note that they are not guaranteed to work across all Windows versions.

Here a list of what is available

rundll32 shell32,Control_RunDLL	<i>Run The Control Panel</i>
rundll32 shell32,Control_RunDLL X	<i>Start applet X of Control Panel</i> ("X" = any CPL filename)
rundll32.exe shell32.dll,Control_RunDLL intl.cpl,,4	<i>Regional setting, Date tab</i>
rundll32 shell32,OpenAs_RunDLL \dir\filename.txt	<i>Open The 'Open With...' Window</i>
rundll32 shell32,ShellAboutA Info-Box	<i>Open 'About Window Window'</i>
rundll32 shell32,Control_RunDLL desk.cpl	<i>Open Display Properties</i>
rundll32 user,cascadechildwindows	<i>Cascade All Windows</i>
rundll32 user,tilechildwindows	<i>Minimize All Child-Windows</i>
rundll32 user,repaintscreen	<i>Refresh Desktop</i>
rundll32 keyboard,disable	<i>Lock The Keyboard</i>
rundll32 mouse,disable	<i>Disable Mouse</i>
rundll32 user,swapmousebutton	<i>Swap Mouse Buttons</i>
rundll32 user,setcursorpos	<i>Set Cursor Position To (0,0)</i>
rundll32 user,wnetconnectdialog	<i>Show 'Map Network Drive' Window</i>
rundll32 user,wnetdisconnectdialog	<i>Show 'Disconnect Network Disk' Window</i>

rundll32 user,disableoemlayer	<i>Display The BSOD (blue screen of death)Wi</i>
rundll32 diskcopy,DiskCopyRunDll	<i>Show Copy Disk Window</i>
rundll32 rnaui.dll,RnaWizard	<i>Run 'Internet Connection Wizard'</i>
rundll32 shell32,SHFormatDrive	<i>Run 'Format Disk (A)' Window</i>
rundll32 shell32,SHExitWindowsEx -1	<i>Cold Restart Of Windows Explorer</i>
rundll32 shell32,SHExitWindowsEx 1	<i>Shut Down Computer</i>
rundll32 shell32,SHExitWindowsEx 0	<i>Logoff Current User</i>
rundll32 shell32,SHExitWindowsEx 2	<i>Windows9x Quick Reboot</i>
rundll32 krnl386.exe,exitkernel	<i>Force Windows 9x To Exit (no confirmation)</i>
rundll32 rnaui.dll,RnaDial "MyConnect"	<i>Run 'Net Connection' Dialog</i>
rundll32 msprint2.dll,RUNDLL_PrintTestPage	<i>Choose &amp; Print Test Page Of Current Print</i>
rundll32 user,SetCareT BlinkTime	<i>Set New Cursor Rate Speed</i>
rundll32 user, SetDoubleClickTime	<i>Set New DblClick Speed (Rate)</i>
rundll32 sysdm.cpl,InstallDevice_Rundll	<i>Hardware installation wizard</i>
rundll32 user,MessageBeep	<i>Default beep sound</i>
rundll32 user32.dll,MessageBeep	<i>Default beep sound (XP)</i>
rundll32 shell32.dll,Control_RunDLL appwiz.cpl	<i>Add/remove programs</i>
rundll32 shell32.dll,Control_RunDLL timedate.cpl,,0	<i>Date/time settings</i>
rundll32 shell32.dll,Control_RunDLL odbc32.cpl	<i>ODBC settings</i>
rundll32.exe url.dll,FileProtocolHandler http:\\www.rgagnon.com	
rundll32.exe url.dll,FileProtocolHandler c:\\mypdf.pdf	<i>Open the associated application</i>
rundll32 amovie.ocx,RunDll /play /close c:\\mymovie.mpg	<i>Play multimedia (movie or sound)</i>
Rundll32.exe powrprof.dll,SetSuspendState Sleep	<i>Put the computer in Sleep mode</i>

#### **Privacy (IE)**

rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 8	<i>Internet temporary files</i>
rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 2	<i>Cookies</i>
rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 1	<i>History</i>
rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 16	<i>Forms Data</i>
rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 32	<i>Passwords</i>
rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 255	<i>Delete everything</i>

The *Windows Fax viewer* is used to view a variety of graphic format like .bmp, .dib, .emf, .gif, .jpeg, .png, .tif or .wmf extensions

```
rundll32.exe shimgvw.dll, ImageView_Fullscreen
"C:\\Documents and Settings\\username\\My Documents\\logo.bmp"
```

In Powerbuilder to launch the rundll32 utility, use the Powerscript Run function.

```
string command
command='rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 8'
Run(command)
```

## Get a list of printers installed

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0001.html>

In this example, we populate a listbox with the printers name

```
string printers[]
int rtn, i, nbPrinters
rtn = RegistryKeys &

("HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Printers", &
    printers)
nbPrinters = UpperBound(printers)
FOR i = 1 TO nbPrinters
    lb_1.addItem(printers[i])
NEXT
```

## Use file association to start an application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0008.html>

Use the external function ShellExecuteA(). The function declaration is :

```
FUNCTION long ShellExecuteA &

    (long hwnd, string lpOperation, &
        string lpFile, string lpParameters, string lpDirectory, &
        integer nShowCmd ) LIBRARY "shell32"
```

PB10+ should use ShellExecuteW instead of ShellExecuteA

From your script, for example, to start the default browser :

```
string ls_Null
long ll_rc

SetNull(ls_Null)
ll_rc = ShellExecuteA &

( handle( this ), "open", "mypahe.html", ls_Null, ls_Null, 1)
```

Possible return codes are:

SE_ERR_FNF	2	// file not found
SE_ERR_PNF	3	// path not found
SE_ERR_ACCESSDENIED	5	// access denied
SE_ERR_OOM	8	// out of memory
SE_ERR_DLLNOTFOUND	32	
SE_ERR_SHARE	26	
SE_ERR_ASSOCINCOMPLETE	27	
SE_ERR_DDETIMEOUT	28	
SE_ERR_DDEFAIL	29	
SE_ERR_DDEBUSY	30	
SE_ERR_NOASSOC	31	

With the next code, it is possible to start an application associated with a specific extension to open a file (with a different extension).

```
[structure nvos_shellexecuteinfo]
long cbSize
long fMask
long hwnd
string lpVerb
string lpFile
string lpParameters
string lpDirectory
long nShow
long hInstApp
long lpIDLList
string lpClass
long hkeyClass
long dwHotKey
long hIcon
long hProcess

[External Function]
FUNCTION long ShellExecuteEx(REF nvos_shellexecuteinfo lpExecInfo)
LIBRARY "shell32.dll" ALIAS FOR ShellExecuteExA

[powerscript]
function boolean _execute
    (readonly string as_file, readonly string as_extension)

CONSTANT long SEE_MASK_CLASSNAME = 1
CONSTANT long SW_NORMAL = 1

string ls_class
long ll_ret
nvos_shellexecuteinfo lnvos_shellexecuteinfo
Inet l_Inet

IF lower(as_extension) = "htm" OR lower(as_extension) = "html" THEN
    // Open html file with HyperlinkToURL
    // So, a new browser is launched
    // (with the code using ShellExecuteEx, it is not sure)
    GetContextService("Internet", l_Inet)
    ll_ret = l_Inet.HyperlinkToURL(as_file)
```

```

IF ll_ret = 1 THEN
    RETURN true
END IF
RETURN false
END IF

// Search for the classname associated with extension
RegistryGet("HKEY_CLASSES_ROOT\." + as_extension, "", ls_class)
IF isNull(ls_class) OR trim(ls_class) = "" THEN
    // The class is not found, try with .txt (why not ?)
    RegistryGet("HKEY_CLASSES_ROOT\.txt", "", ls_class)
END IF

IF isNull(ls_class) OR Trim(ls_class) = "" THEN
    // No class : error
    RETURN false
END IF

lnvos_shellexecuteinfo.cbsize = 60
lnvos_shellexecuteinfo.fMask = SEE_MASK_CLASSNAME // Use classname
lnvos_shellexecuteinfo.hwnd = 0
lnvos_shellexecuteinfo.lpVerb = "open"
lnvos_shellexecuteinfo.lpfFile = as_file
lnvos_shellexecuteinfo.lpClass = ls_class
lnvos_shellexecuteinfo.nShow = SW_NORMAL

ll_ret = ShellExecuteEx(lnvos_shellexecuteinfo)
IF ll_ret = 0 THEN
    // Error
    RETURN false
END IF

RETURN true

```

For example, create the text file c:\test.txt with some lines in.  
Then in Powerbuilder

```

// Open c:\test.txt with Word or Wordpad
of_execute("c:\test.txt", "doc")

// Open c:\test.txt with the default browser
of_execute("c:\test.txt", "htm")

```

This code is useful when you want to open a file which do not have the good extension (for example, a temporary file created with the GetTempFileNameA API has a tmp extension).

Thanks to jc smondack for this tip.

## Shutdown from application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0015.html>

```
[PB external function declaration]
```

```
FUNCTION boolean ExitWindowsEx(ulong uFlags, long dwReserved ) &
```

```
LIBRARY 'user32.dll'
```

```
[Powerscript]
```

```
ulong EWX_LOGOFF    = 0
```

```
ulong EWX_SHUTDOWN = 1
```

```
ulong EWX_REBOOT   = 2
```

```
ExitWindowsEx(EWX_REBOOT, 0)
```

NOTE: While you can shutdown from an application in Win95, you can't with WinNT. You need to call first the AdjustTokenPrivileges API function to grant the current process to right to shutdown the workstation.

```
[structure definitions]
```

```
luid
```

```
    unsignedlong      lowpart
    long             highpart
```

```
luid_and_attributes
```

```
    luid            pluid
    long            attributes
```

```
token_privileges
```

```
    long            privilegecount
    luid_and_attributes  privileges
```

```
[functions declaration]
```

```
Function long OpenProcessToken &
```

```
(long ProcessHandle, long DesiredAccess, ref long TokenHandle) &
```

```
    Library "ADVAPI32.DLL"
```

```
Function long GetCurrentProcess () Library "kernel32"
```

```
Function long LookupPrivilegeValue &
```

```
(string lpSystemName, string lpName, ref LUID lpLUID) &
```

```
    Library "ADVAPI32.DLL" Alias for "LookupPrivilegeValueA"
```

```
Function long AdjustTokenPrivileges &
```

```
(long TokenHandle, long DisableAllPrivileges, &
```

```
    ref TOKEN_PRIVILEGES newstate, long BufferLength, &
```

```

    ref TOKEN_PRIVILEGES PreviousState, ref long ReturnLength) &

    Library "ADVAPI32.DLL"
Function long CloseHandle (long hObject) Library "kernel32"
FUNCTION long ExitWindowsEx(uint Flags, long dwReserved) &

    LIBRARY "User32.dll"

[Powerscript]
Constant string SE_SHUTDOWN_NAME = "SeShutdownPrivilege"

Constant long SE_PRIVILEGE_ENABLED = 2
Constant long TOKEN_ADJUST_PRIVILEGES = 32
Constant long TOKEN_QUERY = 8

CONSTANT long TokenDefaultDacl = 6
CONSTANT long TokenGroups = 2
CONSTANT long TokenImpersonationLevel = 9
CONSTANT long TokenOwner = 4
CONSTANT long TokenPrimaryGroup = 5
CONSTANT long TokenPrivileges = 3
CONSTANT long TokenSource = 7
CONSTANT long TokenStatistics = 10
CONSTANT long TokenType = 8
CONSTANT long TokenUser = 1

CONSTANT INTEGER EWX_LOGOFF = 0
CONSTANT INTEGER EWX_SHUTDOWN = 1
CONSTANT INTEGER EWX_REBOOT = 2
CONSTANT INTEGER EWX_FORCE = 4

// author Philip Salgannik
LUID tLUID
ULong hProcess
Long hToken
TOKEN_PRIVILEGES tTPOld, tTP
Long lTpOld, lR, ll_size
string ls_null
boolean NTEnableShutDown

SetNull(ls_null)

lR = LookupPrivilegeValue(ls_null, SE_SHUTDOWN_NAME, tLUID)

IF (lR <> 0) THEN
    // Get the current process handle:
    hProcess = GetCurrentProcess()
    IF (hProcess <> 0) THEN
        lR = OpenProcessToken &

            (hProcess, TOKEN_ADJUST_PRIVILEGES + TOKEN_QUERY, hToken)
        IF (lR <> 0) THEN

```

```
//Ok we can now adjust the shutdown priviledges:  
tTP.PrivilegeCount = 1  
tTP.Privileges.Attributes = SE_PRIVILEGE_ENABLED  
tTP.Privileges.pLuid.HighPart = tLUID.HighPart  
tTP.Privileges.pLuid.LowPart = tLUID.LowPart  
  
//Now allow this process to shutdown the system:  
ll_size = 16 //sizeof(tTP)  
lR = AdjustTokenPrivileges&  
  
          (hToken, 0, tTP, ll_size, tTPOld, lTpOld)  
IF (lR <> 0) THEN  
    NTEnableShutDown = True  
ELSE  
    MessageBox &  
  
        ("Error", "You do not have the privileges to shutdown")  
END IF  
    CloseHandle(hToken)  
END IF  
END IF  
  
IF NOT NTEnableShutDown THEN RETURN  
  
lR = ExitWindowsEx(ewx_shutdown, 0)  
IF (lR = 0) THEN  
    MessageBox("Error", "ShutdownSystem failed")  
    RETURN  
ELSE  
    RETURN  
END IF
```

## Start the screen saver

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0016.html>

```
/*  
**  WM_SYSCOMMAND 0x0112      274  
**  SC_SCREENSAVE 0xF140      61760  
*/  
send(handle(This), 274, 61760, 0)
```

## Get the CDROM drive letter

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0012.html>

```
[Function declarations]
FUNCTION ulong GetLogicalDrives() LIBRARY "Kernel32.dll"
FUNCTION uint GetDriveType( Ref String as_root_path )
    LIBRARY "kernel32.dll" ALIAS FOR "GetDriveTypeA"

[PB function String of_GetCDRootPath()]

integer    li_ctr
string     ls_root
ulong      lul_drives, lul_rem

lul_drives = GetLogicalDrives()

DO
    lul_rem = MOD(lul_drives, 2)
    IF lul_rem = 1 THEN
        ls_root = Char(li_ctr + 64) + ":\""
        IF GetDriveType(ls_root_path) = 5 THEN
            Return ls_root_path
        END IF
        li_ctr ++
    END IF
    lul_drives /= 2
LOOP UNTIL lul_drives = 0

RETURN ""
```

## Get the user name and the computer name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0002.html>

You need to declare two API calls.

```
FUNCTION long GetComputerNameA &
    (ref string ComputerName, ref ulong BufferLength)
LIBRARY "KERNEL32.DLL"
FUNCTION long GetUserNameA(ref string UserName, ref ulong BufferLength)
LIBRARY "ADVAPI32.DLL"
```

and then

```
long ll_ret
string ls_ComputerName, ls_UserName
ulong BufferLength = 250 // you may need to adjust this. see Note
ls_ComputerName = Space(BufferLength)
```

```

ls_UserName      = Space(BufferLength)

ll_ret = GetComputerNameA(ls_ComputerName, BufferLength)
ll_ret = GetUserNameA(ls_UserName, BufferLength)

```

NOTE : From H. Andersson, "In your example to get the username with the function GetUserNameA you take as bufferlength 250. If you have a longer username (for example administrator) the function doesn't return what we expect. I took 100 as bufferlength and now it works."

## Drive mapping to UNC notation

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0131.html>

To convert a normal paths (N:\PATH) to UNC (\\\\$ERVER\PATH).

```

[local external function declaration]
FUNCTION ulong WNetGetConnectionA &

( ref string drv, ref string unc, ref ulong buf ) &

LIBRARY "mpr.dll"

[powerscript]
string    ls_tmp, ls_unc
Ulong     ll_rc, ll_size

ls_tmp = upper(left(as_path,2))
IF right(ls_tmp,1) <> ":" THEN RETURN as_path

ll_size = 255
ls_unc = Space(ll_size)

ll_rc = WNetGetConnectionA (ls_tmp, ls_unc, ll_size)
IF ll_rc = 2250 THEN
    // prbably local drive
    RETURN as_path
END IF

IF ll_rc <> 0 THEN
    MessageBox("UNC Error", &
        "Error " + string(ll_rc) + " retrieving UNC for " + ls_tmp)
    RETURN as_path
END IF

// Concat and return full path
IF len(as_path) > 2 THEN
    ls_unc = ls_unc + mid(as_path,3)
END IF

RETURN ls_unc

```

## Make a program sleep

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0007.html>

Declare :

```
SUBROUTINE SLEEP (LONG LMILLISEC) LIBRARY "KERNEL32.DLL"
```

and then to sleep (or wait) 2 seconds

```
SLEEP (2000)
```

## Call HtmlHelp

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0019.html>

Declare the following external function

```
FUNCTION long HtmlHelp(long hwnd, string pszFile, &
    long uCommand, long dwData) LIBRARY "hhctrl.ocx" &
    ALIAS FOR "HtmlHelpA"
```

*hwnd* is the handle of calling window (you may set it using handle(this)function)

*pszFile* is the name of help file eg. myApp.chm

*uCommand* usually is 0 or 1

*dwData* is the "numeric" topic (for example, 1005).

NOTE: the hhctrl.ocx must be registered in your system. If you have IE4 installed, it's done already.

To display the HTMLHelp with left panel showing the Table of Content Tab and the right panel the first Help page

```
HtmlHelp( ll_handlewindow, ls_helpfile + ' > main', 0, 0)
```

To display the HTMLHelp with left panel showing the Search Tab and the right panel the first Help page

```
string nullstring  
  
SetNull(nullstring)  
HtmlHelp( ll_handlewindow, ls_helpfile + ' > main', 1, nullstring)
```

To display the HTMLHelp for a specific topic (no left panel).

```
CONSTANT long HH_HELP_CONTEXT = 15
HtmlHelp( lh_handlewindow, ls_helpfile, HH_HELP_CONTEXT, 1005)
```

## Maximize a frame

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0023.html>

Put the following code into the Open event of the Frame Window.

```
CONSTANT Integer WM_SYSCOMMAND = 274
CONSTANT UInt SC_MAXIMIZE = 61488
// 
Send(Handle(This), WM_SYSCOMMAND, SC_MAXIMIZE, 0)
```

other useful constants :

CONSTANT UInt	SC_CLOSE	= 61536
CONSTANT UInt	SC_SCREENSAVE	= 61760
CONSTANT UInt	SC_RESTORE	= 61728
CONSTANT UInt	SC_MINIMIZE	= 61472

Thanks to C. Dadald.

## Make a window popup "on top"

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0026.html>

Declare the following fonction :

```
FUNCTION BOOLEAN SetForegroundWindow( LONG HWND ) LIBRARY "USER32"
```

and

```
long hWnd
hWnd = Handle(w_my_popup)
SetForegroundWindow( HWND )
```

## Make the "hourglass" cursor stay

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0035.html>

Sometime the cursor is reseted after database operations. To make sure the cursor stay in a known state, simply call the following APIs.

```
[FUNCTION DECLARATIONS]
FUNCTION ulong SetCapture(ulong a) LIBRARY "user32.dll"
FUNCTION boolean ReleaseCapture() LIBRARY "user32.dll"

[powerscript]
ulong ll_handle, ll_rc

ll_handle = Handle(this)
ll_rc = SetCapture(ll_handle)
SetPointer(hourglass!)

// some operations

ReleaseCapture()
```

## Move a window without a titlebar

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/pb-0037.html>

In a window

```
[Instance declaration]
CONSTANT uint WM_NCLBUTTONDOWN = 161
CONSTANT uint HTCAPTION = 2

[mousedown event]
Post( Handle( this ), WM_NCLBUTTONDOWN, HTCAPTION, Long( xpos, ypos ) )
```

## Convert ANSI to Oem character set

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/pb-0039.html>

```
[Function declarations]
FUNCTION ulong OemToCharA(ref string source_text, ref string dest_text) &
LIBRARY "user32.DLL"

FUNCTION ulong CharToOemA(ref string source_text, ref string dest_text) &
```

```

LIBRARY "USER32.DLL"

[powerscript]
Integer li_MaxLength
String ls_Source, ls_Dest
Long ll_Ret

ls_Source = "R" + char(130) + "al" // Ascii OEM for "Réal"
li_MaxLength = 64
ls_Dest = Space(li_MaxLength)
ll_Ret = OemToCharA(ls_Source, ls_Dest)
messagebox("OEM - > ANSI", ls_dest)

ls_Source = "Réal"
ll_Ret = CharToOemA(ls_Source, ls_Dest)
messagebox("ANSI - > OEM", ls_dest)

```

## Send keystroke to a control

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails./pbdetails/pb-0041.html>

```

[Function declaration]
SUBROUTINE keybd_event( int bVKey, int bScan, int dwFlags, int dwExtraInfo) &

LIBRARY "user32.dll"

[powerscript]
li_vkey = 65      // the character "A"
sle_1.setfocus() // the target control
keybd_event( li_vkey, 1, 0, 0 )

```

The following will send a backspace to a SingleLineEdit, deleting the last character.

```

[powerscript]
integer li_vkey
integer ll_pos

ll_pos = len(sle_1.Text) + 1
sle_1.selectText(ll_pos,0) // set the cursor at the end
li_vkey = asc(~b)          // backspace
keybd_event( li_vkey, 1, 0, 0 )

```

## Return an ERRORLEVEL to a BAT file

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails./pbdetails/pb-0046.html>

First method:

```
[External FUNCTION declaration]
SUBROUTINE ExitProcess(ulong uExitCode) LIBRARY "kernel32.dll"

[Powerscript]
ExitProcess(50)    // terminate the current application
                   // with ERRORLEVEL 50

[BAT file example]
echo off
start /w myapp.exe
if errorlevel 100 goto 100
if errorlevel 50  goto 50
echo exit code is 0
goto done
:100
echo exit code is 100
goto done
:50
echo exit code is 50
:done
echo done.
```

NOTE: Errorlevel values must range between 0 and 255. Testing this API from inside the Powerbuilder environment (in development mode) will terminate PB itself.

One nice way to implement this is to have a global variable *gul\_rc* with a default value of 0. This variable will be modified to contain a different value if needed. When executing the close event, a call is made to the ExitProcess subroutine passing the current value of *gul\_rc*. Before it may be safer to call the Garbage collector to avoid some memory leaks because ExitProcess() is a rude way to terminate an application. You trigger the Garbage collector by calling the function *GarbageCollect()*.

Second method:

```
[close event of the application]
Message.LongParm = 2    // return 2 as ERRORLEVEL
```

## Change screen resolution

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0049.html>

```
[Local external function declaration]
FUNCTION long ChangeDisplaySettingsA (ref devmode lpst, ulong Flags) &
LIBRARY "USER32.DLL"
```

```

[structure definition, devmode]
character    dmdevicename[32]
integer      dmspecversion
integer      dmdriverversion
integer      dmsize
integer      dmdriverextra
long         dmfields
integer      dmorientation
integer      dmpapersize
integer      dmpaperlength
integer      dmpaperwidth
integer      dmsscale
integer      dmdefaultsource
integer      dmprintquality
integer      dmcolor
integer      dmduplex
integer      dmresolution
integer      dmttoption
integer      dmcollate
character   dmformname[32]
integer      dmlogpixels
long         dbitsperpel
long         dmpelswidth
long         dmpelsheight
long         dmdisplayflags
long         dmdisplayfrequency
long         dmcmmethod
long         dmcmintent
long         dmmediatype
long         dmdithertype
long         dmreserved1
long         dmreserved2

[Instance variable declaration]
Ulong  CDS_FORCE  = 8*16*16*16*16*16*16
long   DM_BITSPERPEL_H = 4*16*16*16*16
long   DM_PELSWIDTH_H = 8*16*16*16*16
long   DM_PELSHEIGHT_H = 16*16*16*16*16
long   DM_DISPLAYFLAGS_H = 2*16*16*16*16*16

[powerscript to switch to 800x600]
devmode dm
long a

dm.dmPelsWidth = 800
dm.dmPelsHeight = 600
dm.dmBitsPerPel = 16
dm.dmFields = DM_PELSWIDTH_H + DM_BITSPERPEL_H
dm.dmSize = 188
a = ChangeDisplaySettingsA(dm, CDS_FORCE)

[powerscript to switch to 1024x768]
devmode dm

```

```

long a

dm.dmPelsWidth = 1024
dm.dmPelsHeight = 768
dm.dmBitsPerPel = 16
dm.dmFields = DM_PELSWIDTH_H + DM_BITSPERPEL_H
dm.dmSize = 188
a = ChangeDisplaySettingsA(dm, CDS_FORCE)

```

Here you can download a [devmode structure](#) and a [test window](#). Just import them in a PBL and run the window.

## Determine what is the decimal or hundred separator at runtime

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0050.html>

When writing or reading data from or to a file, PB uses the current user settings to determine what are the separators used. You may want to take extra steps to ensure that the separator used is a known one (like a "." for decimal). Simply read the registry to get the current separator value and keep it. Modify the Registry for the desired separator. Then do your processing and after put back the original separator value.

```

string ls_decimal, ls_thousand
string ls_regKey = "HKEY_CURRENT_USER\Control Panel\International"
RegistryGet(ls_regKey, "sDecimal", ls_decimal)
RegistryGet(ls_regKey, "sThousand", ls_thousand)

```

You may need to notify Windows applications that a change in the regional settings has been made. You broadcast the WM\_SETTINGCHANGE message then Windows applications will react to that change.

```

[local function declaration]
FUNCTION long SendMessageA &

( long hwnd,  uint Msg,  long wparam,  string lparam) &

LIBRARY "USER32"

[powerscript]
long HWND_BROADCAST = 65535
uint WM_SETTINGCHANGE = 26

String ls_separator
String  ls_regKey = "HKEY_CURRENT_USER\Control Panel\International"

RegistryGet(ls_regkey, "sDecimal", RegString!, ls_separator)

messagebox("current separator", ls_separator)
IF ls_separator"." THEN
// make sure it's a "."

```

```

    RegistrySet(ls_regkey, "sDecimal", RegString!, ".")
    SendMessageA( HWND_BROADCAST, WM_SETTINGCHANGE, 0, 'intl' )
END IF

// dbf format requires a "." as the separator
dw_1.saveas("data.dbf" , dbase3!, true)

// put back the original value
RegistrySet(ls_regkey, "sDecimal", RegString!, ls_separator)
SendMessageA( HWND_BROADCAST, WM_SETTINGCHANGE, 0, 'intl' )

```

NOTE: On Win9x, you need to modify the win.ini (not the registry) file using the ProfileString() function.

A simple way to get the decimal separator without querying the Registry is :

```

IF pos(string(1/2), ".") > 0 THEN
    messagebox("dot", ".")
ELSE
    messagebox("comma", ",")
ENDIF

```

Thanks to C. Frenné for the tip!

You may want to check this [HowTo](#).

## Move the mouse cursor

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0052.html>

```

[structure definition for GetCursorPos function, str_point]
posx  long
posy  long

[local external function declaration]
FUNCTION boolean SetCursorPos(int cx, int cy)  LIBRARY "User32.dll"
FUNCTION boolean GetCursorPos(ref str_point POINT)  LIBRARY "User32.dll"

[clicked event of a button]
int i
str_point lstr_point

GetCursorPos(lstr_point)
FOR i = 0 TO 100
    SetCursorPos(lstr_point.posx + i , lstr_point.posy + i)
NEXT

```

## Disable the MouseWheel ZOOM function

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0063.html>

```
[powerscript - other event]
CONSTANT integer WM_MOUSEWHEEL = 522

IF message.number = WM_MOUSEWHEEL AND &

    KeyDown (KeyControl!) THEN
    message.processed = TRUE
    RETURN 1
END IF
```

## Set the system date/time

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0053.html>

```
[structure definition , str_systemtime]
year      uint
month     uint
dayweek   uint
day       uint
hour      uint
min       uint
sec       uint
millsec   uint

[local external function declaration]
FUNCTION long SetLocalTime(ref str_systemtime lpSystemTime ) &

LIBRARY "kernel32.dll"

[clicked event of a button]
str_systemtime lstr_systemtime

lstr_systemtime.year      = 1999
lstr_systemtime.month     = 1      // January = 1 and so on.
lstr_systemtime.dayweek   = 0      // not used
lstr_systemtime.day       = 3
lstr_systemtime.hour      = 12
lstr_systemtime.min       = 0
lstr_systemtime.sec       = 0
```

```

lstr_systemtime.millsec = 0
SetLocalTime(lstr_systemtime)

```

## Remove the Close item in the system menu.

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0055.html>

```

[external function declaration]
FUNCTION ulong GetSystemMenu( ulong hWnd, boolean bRevert ) &

    LIBRARY "user32.dll"
FUNCTION boolean DeleteMenu( ulong hMenu, uint uPosition, uint uFlags ) &

    LIBRARY "user32.dll"
FUNCTION boolean DrawMenuBar( ulong hWnd ) LIBRARY "user32.dll"
FUNCTION boolean InsertMenuA( ulong hMenu, uint uPosition, uint uFlags, &
    uint uIDNewItem, string lpNewItem ) &

    LIBRARY "user32.dll"

{Powerscript (to remove) }
CONSTANT uint SC_CLOSE = 61536
CONSTANT uint MF_BYCOMMAND = 0

ulong      hWnd, hMenu

hWnd = Handle( this )
hMenu = GetSystemMenu( hWnd, FALSE )
IF hMenu > 0 THEN
    DeleteMenu( hMenu, SC_CLOSE , MF_BYCOMMAND)
    DrawMenuBar( hWnd )
END IF

{Powerscript (to insert) }
InsertMenuA &

( hMenu, SC_CLOSE , MF_BYCOMMAND, SC_CLOSE, "&Close~tAlt+F4" )

```

NOTE: By removing the Close item in the system menu, the top-right 'X' is disable.

While the Close item is gone, the user still can do an ALT-F4 to close the window. To trap ALT-F4 you need to process the SC\_CLOSE Windows message. Here how it can be done.

```

[ue_syscommand mapped to pbm_syscommand]
CONSTANT unsignedlong SC_CLOSE = 61536

```

```

IF commandType = SC_CLOSE THEN
    RETURN 1
END IF

```

NOTE : Other possible values : 61472 (Minimize), 61458 (Title Bar/Move), 61728 (Restore)

## Flash a Window Title bar

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0096.html>

```

[structure s_flashinfo]
cbSize     unsignedlong
hwnd       unsignedlong
dwFlags    unsignedlong
uCount     unsignedlong
dwTimeout  unsignedlong

[external function declaration]
FUNCTION boolean FlashWindowEx(REF s_flashinfo str_flashinfo) &

LIBRARY "user32.dll"

[powerscript]
CONSTANT unsignedlong FLASHW_STOP = 0      // Stop flashing
CONSTANT unsignedlong FLASHW_CAPTION = 1    // Flash the window caption
CONSTANT unsignedlong FLASHW_TRAY = 2        // Flash the taskbar button.
// Flash both the window caption and taskbar button.
CONSTANT unsignedlong FLASHW_ALL = 3
// Flash continuously, until the FLASHW_STOP flag is set.
CONSTANT unsignedlong FLASHW_TIMER = 4
// Flash continuously until the window comes to the foreground
CONSTANT unsignedlong FLASHW_TIMERNOFG = 12

ulong ll_win
s_flashinfo lstr_flashinfo

lstr_flashinfo.cbSize = 20
lstr_flashinfo.hwnd = Handle(this) // handle(parent) if from a control
lstr_flashinfo.dwFlags = FLASHW_ALL
lstr_flashinfo.uCount = 10          // 10 times
lstr_flashinfo.dwTimeout = 0      // speed in ms, 0 default blink cursor rate

FLASHWINDOWEX(LSTR_FLASHINFO)

```

The FlashWindowEx() API is only available on Win98 or WinNT/Win2000.

On Win95 or NT4, use this API instead. Call it in a loop or from a timer event to toggle the Window title bar.

```
[external function declaration]
FUNCTION boolean FlashWindow(ulong hndl boolean flash) &
LIBRARY "user32.dll"
```

## Detect if an application is already running

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0064.html>

```
[external function declaration]
FUNCTION ulong CreateMutexA &

(ulong lpMutexAttributes, boolean bInitialOwner, REF string lpszName) &
LIBRARY "kernel32.dll"
FUNCTION long GetLastError() LIBRARY "kernel32.dll"

[powerscript]
// boolean of_IsRunning()
//
//      IF of_isRunning THEN
//          MessageBox("Oups", "already running!")
//      END IF
//

constant ulong ERROR_ALREADY_EXISTS = 183
constant ulong SUCCESSFUL_EXECUTION = 0

ulong lul_mutex
ulong lpsa
ulong lul_last_error
boolean lb_ret = FALSE

IF NOT (Handle(GetApplication()) = 0) THEN
    lul_mutex = CreateMutexA(lpsa, FALSE, as_appname)
    lul_last_error = GetLastError()
    lb_ret = NOT (lul_last_error = SUCCESSFUL_EXECUTION)
END IF

RETURN LB_RET
```

## Retrieve error from calling a Win API

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0118.html>

If a Win API call fails for any reason, a return code is returned. Habitually, an error message is available. You can get it by calling the FormatMessage() function.

```
[local external function declaration]
FUNCTION long GetLastError() LIBRARY "kernel32" ALIAS FOR "GetLastError"
FUNCTION long FormatMessage  &
(Long dwFlags ,ref Any lpSource , Long dwMessageId , &
Long dwLanguageId , ref String lpBuffer , &
Long nSize , Long Arguments) LIBRARY "kernel32" &
ALIAS FOR "FormatMessageA"
```

In the following example, we call the ShellExecute API giving it a non-existent filename. Then we can get the error message generated by the Windows API call.

```
[local external function declaration]
FUNCTION long ShellExecuteA( long hwnd, string lpOperation, &
string lpFile, string lpParameters, string lpDirectory, &
integer nShowCmd ) LIBRARY "SHELL32"

string ls_Null
long ll_rc
string ls_err_str
long ll_last_error
Any temp
CONSTANT long FORMAT_MESSAGE_FROM_SYSTEM = 4096

SetNull(ls_Null)
// try to execute a non-existent filename.
ll_rc = ShellExecuteA( Handle( This ), "open", &
"MyPage.xyz", ls_Null, ls_Null, 1)

IF ll_rc > 1 THEN
temp = 0
ll_last_error = GetLastError()
ls_err_str = Fill(Char(0),255)
FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, temp, ll_last_error,&
0, ref ls_err_str, 255, 0)
MessageBox("error", ls_err_str)
END IF
```

## Use the default MAPI profile

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0119.html>

```
// Win NT
MailSession lms_mSes

lms_mSes = CREATE MailSession
li_rc = RegistryGet( &

    "HKEY_CURRENT_USER\Software\Microsoft\Windows NT\" &
    + "CurrentVersion\Windows Messaging Subsystem\Profiles" , &
    "DefaultProfile", RegString!, ls_profile )
// Win9x,
// "HKCU\Software\Microsoft\Windows Messaging Subsystem\Profiles"

IF li_rc = FAILURE THEN
    // no registry entries for default MAPI profile
    RETURN -1
END IF
lms_mSes.MailLogon(ls_profile, "", mailNewSession!)
```

Calling the MailLogon change the current directory so it's a good idea to keep the current directory before making the call and restore it after.

Using the PFC, this should like this :

```
n_cst_filesrv lnv_fsrv

f_SetFileSrv(lnv_fsrv, TRUE)
ls_path = lnv_fsrv.of_GetCurrentDirectory()

// MAPI stuff

lnv_fsrv.of_ChangeDirectory(ls_path)
f_SetFileSrv(lnv_fsrv, FALSE)
```

## Set and retrieve the executable version

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0120.html>

On Windows, the properties tab on an executable can show a version number. By default, an executable

produced by Powerbuilder shows the Powerbuilder version number not the version of the executable itself.

Here a way to embed your own version number. By reading it back, it's possible to validate if the executable is the current one against a special table in your application's database for example.

---

You need a special utility to change the version of the executable. A freeware utility to do that can be downloaded from <http://www.elphin.com/downloads/stampver/>.

Create a response file to contains you version number with the right formatting (in this example, only the first 3 numbers are used).

```
;StampVer information file (myapp.inf)
FileVersion=1.0.2.0
FileFormat=%02a.%02b.%02c
```

Now you can the version number with

```
stampver -v"myapp.inf" myapp.exe
```

Showing the properties/version tab for this executable will show your version number and not the Sybase one now.

This freeware can deal only with the file version, not the product information or any other infos shown on properties/version tab. To change these values, look at the [Crane computing](#) site, they have product that can do that with a nice gui interface.

---

To retrieve the version from an executable :

```
[local external function declaration]
FUNCTION ulong GetFileVersionInfoSizeA &

( REF string lpFilename, REF ulong lpdwHandle ) &
LIBRARY "version.dll"

FUNCTION integer GetFileVersionInfoA &

( REF string lpFilename, REF ulong lpdwHandle, ulong dwLen, &
REF string lpData ) &
LIBRARY "version.dll"

FUNCTION boolean VerQueryValueA &

( REF string lpBlock, string lpSubBlock, REF long lpBuffer, &
REF uint puLen ) &
LIBRARY "version.dll"

SUBROUTINE CopyMemory &
```

```

( REF string d, long s, long l )  &
LIBRARY "kernel32.dll"  &
ALIAS FOR RtlMoveMemory

[powerscript]
ulong dwHandle, dwLength
string ls_Buff, ls_key, ls_versioninfo
uint lui_length
long ll_pointer
string as_filename = "d:\dev\pb6\myapp.exe"
integer li_rc

dwLength = GetFileVersionInfoSizeA( as_filename, dwHandle )
IF dwLength <= 0 THEN
    RETURN
END IF

ls_Buff = Space( dwLength )

li_rc = GetFileVersionInfoA( as_filename, dwHandle, dwLength, ls_Buff )

IF li_rc = 0 THEN
    RETURN
END IF

// the strange numbers below represents the country and language
// of the version ressource.
ls_key = "\StringFileInfo\040904e4\FileVersion"
IF NOT VerQueryValueA( ls_buff, ls_key, ll_pointer, lui_length ) OR &

    lui_length <= 0 THEN
        ls_versioninfo = "?"
ELSE
    ls_versioninfo = Space( lui_length )
    CopyMemory( ls_versioninfo, ll_pointer, lui_length )
END IF

Messagebox("version", ls_versioninfo)

```

## Get the application PID

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0122.html>

The PID (process identifier) is a number assigned by the OS for a running process. You can view the allowed PIDs through the Task List on NT. This number can be useful to kill a process with a suitable utility.

```
[local external function declaration]
FUNCTION long GetCurrentProcessId ( ) LIBRARY "kernel32.dll"

[powerscript]
long ll_pid
ll_pid = GetCurrentProcessId()
messagebox("PID" , string(ll_pid))
```

## Create and use a C DLL (MSVC6)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0123.html>

1. With VC++ v6 , create a new *Win32 Dynamic DLL* project and name it : *howto*
2. Specify that it is a *DLL that exports some symbols*
3. Edit the *howto.cpp* file.

Add the following function :

```
HOWTO_API char * __stdcall sayhello(void)
{
    return "hello";
}
```

4. Edit the *howto.h* file.

Add the following line :

```
HOWTO_API CHAR * __STDCALL sayhello(VOID);
```

5. In the Project settings for *Win32 release*.

Make sure that you are *not using the MFC* in *General Tab*.

In the *Link tab*, category *General*, check the option *Generate MAP file*.

Click the OK button.

6. From the Build menu, set the active configuration to *Win 32 release*.

Build the project.

7. Open the file *howto.map* located in the *Release* directory.

Find the *sayhello* entry and note the "decorated name".

In this example, it's *?sayhello@@YGPADXZ*

8. Click on *Source files* (in the treeview) and from Menu choose *New*.

Select *Text file*, check *Add to the project* and give *howto.def* as the name.

Enter the following lines in the new *howto.def* file :

```
; DEF file for the Howto DLL
EXPORTS
sayhello =?sayhello@@YGPADXZ
```

note: this will add `/def:".\howto.def"` to the Project options (you can see these options in Project settings).

9. Save everything and build the DLL
10. Place the `howto.dll` (located in the *Release* directory) in the same directory as your PBL.
11. To use the DLL from PB, add the *Local external function declaration* :

```
function string sayhello() library "howto.dll"
```

12. In Powerscript

```
messagebox("", string(sayhello()))
```

13. A more useful example is to create a `bitwiseAnd` function since this is not supported in Powerscript unless you use complicated string manipulations which are very time consuming.

Add the following declaration in the `howto.h` file :

```
HOWTO_API DWORD __STDCALL bitwiseAnd( DWORD , DWORD );
```

14. Add the following function to the `howto.cpp`

```
HOWTO_API DWORD __stdcall bitwiseAnd( DWORD arg1, DWORD arg2)
{
    return ( arg1 & arg2 );
}
```

15. Rebuild and determine from the `howto.map` file the decorated name for the `bitwiseAnd` function.

On my system, it's `?bitwiseAnd@@YGK@Z`. So now the `howto.def` file should look like :

```
; DEF file for the Howto DLL
EXPORTS
sayhello =?sayhello@@YGPADXZ
bitwiseAnd = ?bitwiseAnd@@YGK@Z
```

16. Rebuild everything with the new `howto.def`.

17. From Powerscript, you can use your new function with :

```
[local external function declaration]
FUNCTION ulong bitwiseAnd(ulong a, ulong b) LIBRARY "howto.dll"

[powerscript]
ulong a = 3
ulong b = 2
ulong c

c = bitwiseAnd( a , b )
messagebox("", string(c)) // should be 2
```

Where returning a string to PB from a DLL, you should always allocate some spaces from PB. In the following example, the `sayhello()` will return "hello" in the `sayit` variable passed by reference. Before the actual call, the PB variable is initialized with enough spaces.

```
[local external function declaration]
SUBROUTINE sayhello(ref string sayit) LIBRARY "howto.dll"

[Powerscript]
string ls_hello

ls_hello = space(20)
```

```
sayhello(ls_hello)
MessageBox("", ls_hello)
```

## Use Windows Resources from a DLL

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0127.html>

Let's create 2 DLLs containing StringTables for a particular language. We create these DLLs with VC++.

First, the French DLL

1. Select **File - New**
2. Choose **Win32 Dynamic Link** with the name **resFr**
3. It's a **Simple DLL project**
4. Select **Insert - Resource**, and **StringTable - New**
5. The Strings are

```
ST_1    Ligne un
ST_2    Ligne deux
```

6. Save the resource as **res\_fr.rc** and add that file to the project source folder.
7. Select **Build - set Active Configuration to Win32 Release**
8. Select **Build resFr.dll**

The English DLL

1. Select **File - New**
2. Choose **Win32 Dynamic Link** with the name **resEn**
3. It's a **Simple DLL project**
4. Select **Insert - Resource**, and **StringTable - New**
5. The Strings are

```
ST_1    Line one
ST_2    Line two
```

6. Save the resource as **res\_en.rc** and add that file to the project source folder.
7. Select **Build - set Active Configuration to Win32 Release**
8. Select **Build resEn.dll**

To use these resources from PowerScript, you need the following local external function declarations.

```
FUNCTION  ulong LoadLibraryA (string lpLibFileName) LIBRARY "kernel32.dll"
FUNCTION  boolean FreeLibrary (ulong hLibModule) LIBRARY "Kernel32.dll"
FUNCTION  int   LoadStringA(ulong hInstance, uint Uid, ref string &
lpBuffer, int nBufferMax) LIBRARY "user32.dll"
```

Then from your script

```

ulong lul_res
string ls_temp
constant uint ST_1 = 1
constant uint ST_2 = 2

lul_res = LoadLibraryA("resFr.dll")
ls_temp = Space(255)
LoadStringA(lul_res, ST_1, ls_temp, 255)
MessageBox("The French resource ST_1", ls_temp)
FreeLibrary(lul_res)

lul_res = LoadLibraryA("resEn.dll")
ls_temp = Space(255)
LoadStringA(lul_res, ST_2, ls_temp, 255)
MessageBox("The English resource ST_2", ls_temp)
FreeLibrary(lul_res)

```

#### From a static text constructor

```

ulong lul_res
string ls_temp
constant uint ST_1 = 1

lul_res = LoadLibraryA("resEn.dll")
ls_temp = Space(255)
LoadStringA(lul_res, ST_1, ls_temp, 255)
this.text = ls_temp
FreeLibrary(lul_res)

```

## Get the IP address

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0132.html>

#### PB 9 or less

```

[Structure]
str_wsadata
    unsignedinteger version
    unsignedinteger highversion
    character description[257]
    character systemstatus[129]
    nsignedinteger maxsockets
    unsignedinteger maxupddg
    string vendorinfo

[External function]

function int WSAStartup (uint UIVerionrequested, ref str_wsadata lpWSAdata)
library "wsock32.DLL"

```

```

function int WSACleanup() library "wsock32.DLL"
function int WSAGetLastError() library "wsock32.DLL"
function int gethostname(ref string name, int namelen) library
"wsock32.DLL"
function string GetHost(string lpszhost, ref blob lpszaddress) library
"pbws32.dll"

[Powerscript]

String ls_ip, ls_host
Blob{4} lb_host
Integer li_version, li_rc
str_wsadata lstr_wsadata

ls_host = Space(128)
li_version = 257

If WSAStartup(li_version, lstr_wsadata) = 0 Then
  If GetHostName(ls_host, Len(ls_host)) <0 Then
    li_rc = WSAGetLastError()
  Else
    GetHost(ls_host, lb_host)
    ls_ip = String(Asc(String(BlobMid(lb_host, 1, 1)))) + "."
    ls_ip += String(Asc(String(BlobMid(lb_host, 2, 1)))) + "."
    ls_ip += String(Asc(String(BlobMid(lb_host, 3, 1)))) + "."
    ls_ip += String(Asc(String(BlobMid(lb_host, 4, 1))))
    li_rc = 0
  End If
  MessageBox("My IP", ls_ip)
Else
  li_rc = WSAGetLastError()
End If

WSACleanup()

```

NOTE : The **pbws32.dll** can be downloaded [here](#). This dll was originally distributed with the FTP example in the PB5 Application Gallery (also included in the ZIP).

### PB10 or more.

Small fixes are required since Powerbuilder is now unicode-based and the API used are ANSI-based.

```

[Structure]

... same ...

[External function]
function int WSAStartup (uint UIVerionrequested, ref str_wsadata lpWSAdata) &

  library "wsock32.DLL" alias for "WSAStartup;ansi"
function int WSACleanup() library "wsock32.DLL"
function int WSAGetLastError() library "wsock32.DLL"
function int gethostname(ref string name, int namelen) &

```

```

library "wsock32.DLL" alias for "gethostname;Ansi"
function string GetHost(string lpszhost,ref blob lpszaddress) &

library "pbws32.dll" alias for "GetHost;Ansi"

[Powerscript]

...
ls_ip = String(Asc(String(BlobMid(lb_host, 1, 1),Encodingansi!))) + "."
ls_ip += String(Asc(String(BlobMid(lb_host, 2, 1),Encodingansi!))) + "."
ls_ip += String(Asc(String(BlobMid(lb_host, 3, 1),Encodingansi!))) + "."
ls_ip += String(Asc(String(BlobMid(lb_host, 4, 1),Encodingansi!)))
...

```

---

If you have a valid Visual Studio installation, you have a special ActiveX that can be used. Drop on a window the OLE control called "Microsoft Winsock Control (MSWINSCK.OCX)" and then from Powerscript :

```

String ls_ip

ls_ip = ole_1.object.LocalIP
MESSAGEBOX("IP", LS_IP)

```

To get the IP address from a PB component hosted by EA Server, see this [HowTo](#)

---

## Use Windows Debug API

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/pb-0152.html>

Windows provides API to debug applications and PB can easily send messages to the debugging console

```

[local external routine declaration]
SUBROUTINE OutputDebugStringA (String lpszOutputString) &

LIBRARY "kernel32.dll"

// FOR PB10 :
// SUBROUTINE OutputDebugStringA (String lpszOutputString) &

// LIBRARY "kernel32.dll" ALIAS FOR "OutputDebugStringA;Ansi"

[powerscript]
// if there is a Windows debugging console opened
// then the time will be displayed
OutputDebugStringA( String(Now(), "hh:mm") )

```

A (free) Windows debugging console can be downloaded from [SysInternals \(Microsoft\)](#) Web site.

See this related [HowTo](#).

## Animate a Window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0169.html>

Starting with Win98, a new API is available to add a special effect to your application. The AnimateWindow() is very easy to use, you simply need to pass your window's handle, a delay and some flags to specify the desired effect. These effects are designed to enhance the opening or closing of a window.

```
[local function declaration]
FUNCTION boolean AnimateWindow( long lhWnd, long lTm, long lFlags ) &

LIBRARY 'user32'

[instance variable]
CONSTANT LONG AW_HOR_POSITIVE = 1
CONSTANT LONG AW_HOR_NEGATIVE = 2
CONSTANT LONG AW_VER_POSITIVE = 4
CONSTANT LONG AW_VER_NEGATIVE = 8
CONSTANT LONG AW_CENTER = 16
CONSTANT LONG AW_HIDE = 65536
CONSTANT LONG AW_ACTIVATE = 131072
CONSTANT LONG AW_SLIDE = 262144
CONSTANT LONG AW_BLEND = 524288

[powerscript, open event]
// slide right to left
AnimateWindow ( Handle( this ),500,AW_HOR_NEGATIVE)

// slide left to right
AnimateWindow ( Handle( this ),500,AW_HOR_POSITIVE)

// slide top to bottom
AnimateWindow ( Handle( this ),500,AW_VER_POSITIVE)

// slide bottom to top
AnimateWindow ( Handle( this ),500,AW_VER_NEGATIVE)

// from center expand
AnimateWindow ( Handle( this ),500,AW_CENTER)

// reveal diagonally
AnimateWindow ( Handle( this ),500,AW_VER_NEGATIVE + AW_HOR_NEGATIVE)
```

Here some notes about the flags (from MSDN)

AW_SLIDE	Uses slide animation. By default, roll animation is used. This flag is ignored when used with the AW_CENTER flag.
AW_ACTIVATE	Activates the window. Do not use this flag with AW_HIDE.
AW_BLEND	Uses a fade effect. This flag can be used only if hwnd is a top-level window.
AW_HIDE	Hides the window. By default, the window is shown.
AW_CENTER	Makes the window appear to collapse inward if the AW_HIDE flag is used or expand outward if the AW_HIDE flag is not used.
AW_HOR_POSITIVE	Animates the window from left to right. This flag can be used with roll or slide animation. It is ignored when used with the AW_CENTER flag.
AW_HOR_NEGATIVE	Animates the window from right to left. This flag can be used with roll or slide animation. It is ignored when used with the AW_CENTER flag.
AW_VER_POSITIVE	Animates the window from top to bottom. This flag can be used with roll or slide animation. It is ignored when used with the AW_CENTER flag.
AW_VER_NEGATIVE	Animates the window from bottom to top. This flag can be used with roll or slide animation. It is ignored when used with the AW_CENTER flag.

## Use Microsoft Crypto API

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0170.html>

With almost all Windows installation, the Microsoft Crypto API is available.

CryptoAPI 1.0 is provided through Microsoft Windows NT 4.0 and Microsoft Internet Explorer 3.0 and later. CryptoAPI 1.0 will also ship with the Windows 95 update.

Microsoft provides a separate COM object to make it easy to exploit this API from VBScript or Powerbuilder. But you need to installed the COM object before using it. This How-to will show you how to call directly the Crypto DLL.

The *n\_cst\_crypto* object can encrypt/decrypt a string based on a given key. This can be used to encrypt user/password entries in INI file for example.

Based on this [Visual Basic](#) example, the PB7 PBL containing the *n\_cst\_crypto* object can be download from [here](#).

Many thanks to Martyn Bannister for VB to PB development.

To encrypt a string

```
n_cst_crypto lnv_crypt
string ls_encrypted

lnv_crypt = CREATE n_cst_crypto
ls_encrypted = lnv_crypt.EncryptData("my sensitive data" , "SecretKey")
DESTROY lnv_crypt
```

To decrypt a string

```
n_cst_crypto lnv_crypt
string ls_decrypted

lnv_crypt = CREATE n_cst_crypto
ls_decrypted = lnv_crypt.DecryptData(ls_encrypted , "SecretKey")
DESTROY lnv_crypt
```

Check this [How-to](#) for a powerscript-only.

---

J.C. de Souza Ribeiro wrote :

I tried to use your "PB7 PBL containing the n\_cst\_crypto object" to encrypt a string of 59 numeric digits, using the MD5 algorithm, but everytime I run it, using different values, I get strings of different lengths and I was expecting always a string of 32 bytes.

My answer was

The original VB algorithm is making a great deal to make sure that there is no tab/cr/lf characters in the result string and do encryption (with a counter) again if there are present. I suspect this is why the length is varying.

After some research, he sent me a function using the same API to solve his problem. The main idea to convert the result into hexadecimal format to eliminate the control characters.

```
/*
External Function Definitions
FUNCTION Boolean CryptAcquireContextA (ref ulong hProv, &
                                         ref string pszContainer, &
                                         ref string pszProvider, ulong dwProvType, &
                                         ulong dwFlags) &

LIBRARY "advapi32.dll"

FUNCTION Boolean CryptReleaseContext (ulong hProv, ulong dwFlags) &

LIBRARY "advapi32.dll"

FUNCTION Boolean CryptCreateHash (ulong hProv, uint Algid, ulong hKey, &
```

```

        ulong dwFlags, ref ulong phHash) &

LIBRARY "advapi32.dll"

FUNCTION Boolean CryptHashData (ulong hHash, ref string pbData, &
        ulong dwDataLen, ulong dwFlags) &

LIBRARY "advapi32.dll"

FUNCTION Boolean CryptDestroyHash (ulong hHash) &

LIBRARY "advapi32.dll"

FUNCTION Boolean CryptGetHashParam (ulong hHash, ulong dwParam, &
        ref blob pbData, &
        ref ulong pdwDataLen, ulong dwFlags) &

LIBRARY "advapi32.dll"

FUNCTION Ulong GetLastError () Library "kernel32.dll"
*/



// Constants
CONSTANT ULONG PROV_RSA_FULL = 1
CONSTANT ULONG CRYPT_VERIFYCONTEXT = 4026531840 // 0xF0000000
CONSTANT ULONG CALG_MD5 = 32771 // 4<<13 | 0 | 3
CONSTANT ULONG HP_HASHVAL = 2 // 0x0002


public function string of_md5 (string as_text);
    // Calculate the MD5 message digest hash of a string
    // Using the Windows Crypto API

    ulong MD5LEN = 16
    ulong hProv // provider handle
    ulong hHash // hash object handle
    ulong err_number
    String s_result, s_null
    Integer i, l, r, b
    Blob{16} bl_hash
    Blob{1} bl_byte

    SetNull (s_null)
    ulong cbHash = 0
    CHAR HexDigits[0 TO 15] = &
        {'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'}

    //Get handle to the crypto provider
    IF NOT CryptAcquireContextA&

```

```

(hProv, s_null, s_null, PROV_RSA_FULL, CRYPT_VERIFYCONTEXT) &

THEN
err_number = GetLastError()
return 'acquire context failed ' + String (err_number)
END IF

// Create the hash object
IF NOT CryptCreateHash(hProv, CALG_MD5, 0, 0, hHash) THEN
err_number = GetLastError()
CryptReleaseContext(hProv, 0)
return 'create hash failed ' + String (err_number)
END IF

// Add the input to the hash
IF NOT CryptHashData(hHash, as_text, Len(as_text), 0) THEN
err_number = GetLastError()
CryptDestroyHash(hHash)
CryptReleaseContext(hProv, 0)
return 'hashdata failed ' + String (err_number)
END IF

// Get the hash value and convert it to readable characters
cbHash = MD5LEN
IF (CryptGetHashParam(hHash, HP_HASHVAL, bl_hash, cbHash, 0)) THEN
FOR i = 1 TO 16
    bl_byte = BlobMid (bl_hash, i, 1)
    b = Asc (String(bl_byte))
    r = Mod (b, 16) // right 4 bits
    l = b / 16        // left 4 bits
    s_result += HexDigits [l] + HexDigits [r]
NEXT
ELSE
    err_number = GetLastError()
    return 'gethashparam failed ' + String (err_number)
END IF

// clean up and return
CryptDestroyHash(hHash)
CryptReleaseContext(hProv, 0)

return s_result

```

---

Since PB10 is Unicode, you need to specify to the String() function to use ANSI.

```

[PB7/8/9]
    b = Asc (String(bl_byte))
[PB10]
    b = AscA(String(bl_byte, EncodingANSI!))

```

thanks to ramo76 for the PB10 AscA bug fix

---

Another fix for PB10.5 from Thomas Mathys.

I took your code to encode password by MD5 in a powerbuilder 10.5 application, but I saw the result was different than what I obtained with others tools (Oracle, Javascript example,...), if I encoded more than one character.

I searched a lot, and finally found a solution: you must pass your text as a blob, and not as a string, to the CryptHashData function. You got to change the prototype and your powerscript must be changed like this:

```
...
// Add the input to the hash
Blob bl_text
bl_text = Blob(as_text, EncodingANSI!)
IF NOT CryptHashData(hHash, bl_text, Len(bl_text), 0) THEN
...

```

---

A complete Powerbuilder object is available here :  
[http://www.topwizprogramming.com/freecode\\_crypto.html](http://www.topwizprogramming.com/freecode_crypto.html)

---

If you want to generate SHA1 hash then your DBMS may help you.

For Oracle 10g, it's something like :

```
SELECT DBMS_CRYPTO.HASH (utl_raw.cast_to_raw('Tester12'), :sh1) FROM dual;
```

For MS SQL Server, it's the HashBytes function, see  
<http://msdn.microsoft.com/en-us/library/ms174415.aspx>

## Scroll a Window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0175.html>

All you have to do is to send a particular message to Windows.

```
Window message:
WM_SCROLL      276    // Horizontal scroll
WM_VSCROLL     277    // Vertical scroll

Parameters:
SB_LINEUP      0      // Scrolls one line up.
SB_LINELEFT    0
SB_LINEDOWN    1      // Scrolls one line down.
SB_LINERIGHT   1
SB_PAGEUP      2      // Scrolls one page up.
SB_PAGELEFT    2
SB_PAGEDOWN    3      // Scrolls one page down.
SB_PAGERIGHT   3
SB_TOP         6      // Scrolls to the upper left.
```

```

SB_LEFT      6
SB_BOTTOM    7      // Scrolls to the lower right.
SB_RIGHT     7
SB_ENDSCROLL 8      // Ends scroll.

[powerscript]
// 
// Scroll Window one page down
// WM_VSCROLL = 277
// SB_PAGEDOWN = 3
//
Send(Handle(this), 277, 3, 0)

```

## Get OS version

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0180.html>

You can't rely on the PB Environment Object because it doesn't return enough details. For on W2K system, the Environment returns NT as the operating system. A better way is to call directly the Win API to query the OS version.

```

[local external function]
FUNCTION ulong GetVersionExA( REF str_osversioninfo lpVersionInfo ) &
LIBRARY "kernel32.dll"

```

the required structure

```

[str_osversioninfo]
ulong dwOSVersionInfoSize
ulong dwmajorversion
ulong dwminorversion
ulong dwbuildnumber
ulong dwplatformid
CHARACTER SZCSDVERION[128]

```

the possible values

dwMajorVersion	
Windows 95:	4
Windows 98	4
Windows ME	4
Windows NT 3.51	3
Windows NT 4	4
Windows 2000	5
Windows XP	5

dwMinorVersion

Windows 95	0
Windows 98	10
Windows ME	90
Windows NT 3.51	51
Windows NT 4	0
Windows 2000	0
WINDOWS XP	1

To distinguish between 95 and NT, you also need to check the dwPlatformId value

VER_PLATFORM_WIN32s	0
VER_PLATFORM_WIN32_WINDOWS	1 // WIN95
VER_PLATFORM_WIN32_NT	2 // NT

and from Powerscript, for example

```
str_osversioninfo lstr_osver

lstr_osver.dwosversioninfosize = 148
GetVersionExA( lstr_osver )

IF (lstr_osver.dwmajorversion = 5 AND lstr_osver.dwminorversion = 1) THEN
    MessageBox("", "Running on XP");
END IF
```

## Retrieve the Regional Setting w/o using the Registry

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/.pbdetails/pb-0181.html>

```
[local external functions]
FUNCTION int GetLocaleInfoA(ulong locale, ulong lctype, REF string data, int size) &

    LIBRARY "kernel32.dll"
FUNCTION ulong GetSystemDefaultLCID() LIBRARY "kernel32.dll"

[powerscript]
CONSTANT long LOCALE_ILANGUAGE = 1          ' LANGID in hexadecimal digits
CONSTANT long LOCALE_SLANGUAGE = 2            ' Full localized name of the language
CONSTANT long LOCALE_SENGLANGUAGE = 4097      ' Full English U.S. name of the language ISO Stan
CONSTANT long LOCALE_SABBREVLANGNAME = 3        ' Abbreviated name of the language, ISO Standard
CONSTANT long LOCALE_SNATIVELANGNAME = 4        ' Native name of the language
CONSTANT long LOCALE_ICOUNTRY = 5              ' Country code, based on international phone code
CONSTANT long LOCALE_SCOUNTRY = 6              ' The full localized name of the country.
CONSTANT long LOCALE_SENGCOUNTRY = 4098        ' The full English U.S. name of the country.
CONSTANT long LOCALE_SABBREVTRYNAME = 7         ' Abbreviated name of the country ISO Standard 31
```

```

CONSTANT long LOCALE_SNATIVECTRYNAME = 8      ' Native name of the country.
CONSTANT long LOCALE_IDEFAULTLANGUAGE = 9       ' LANGID for the principal language spoken in this
CONSTANT long LOCALE_IDEFAULTCOUNTRY = 10        ' Country code for the principal country in this
CONSTANT long LOCALE_IDEFAULTCODEPAGE = 11        ' OEM code page associated with the country.
CONSTANT long LOCALE_SLIST = 12                  ' Characters used to separate list items.
CONSTANT long LOCALE_IMEASURE = 13                ' 0 for metric system (S.I.) and 1 for the U.S.
CONSTANT long LOCALE_SDECIMAL = 14                ' decimal separator
CONSTANT long LOCALE_STHOUSAND = 15              ' thousand separator
CONSTANT long LOCALE_SGROUPING = 16              ' Sizes for each group of digits to the left of the
                                                ' number of fractional digits
CONSTANT long LOCALE_IDIGITS = 17                ' 0 means use no leading zeros; 1 means use leading
                                                ' zeros
CONSTANT long LOCALE_ILZERO = 18                 ' local monetary symbol
CONSTANT long LOCALE_SNATIVEDIGITS = 19          ' International monetary symbol ISO 4217.
CONSTANT long LOCALE_SCURRENCY = 20               ' local monetary symbol
CONSTANT long LOCALE_SINTLSYMBOL = 21             ' International monetary symbol ISO 4217.
CONSTANT long LOCALE_SMONDECIMALSEP = 22          ' monetary decimal separator
CONSTANT long LOCALE_SMONTHOUSANDSEP = 23         ' monetary thousand separator
CONSTANT long LOCALE_SMONGROUPING = 24            ' monetary grouping
CONSTANT long LOCALE_ICURRDIGITS = 25             '# local monetary digits
CONSTANT long LOCALE_IINTLCURRDIGITS = 26          '# intl monetary digits
CONSTANT long LOCALE_ICURRENCY = 27                ' positive currency mode
CONSTANT long LOCALE_INEGCURR = 28                ' negative currency mode
CONSTANT long LOCALE_SDATE = 29                   ' date separator
CONSTANT long LOCALE_STIME = 30                   ' time separator
CONSTANT long LOCALE_SSHORTDATE = 31               ' short date format string
CONSTANT long LOCALE_SLONGDATE = 32               ' long date format string
CONSTANT long LOCALE_STIMEFORMAT = 4099            ' time format string
CONSTANT long LOCALE_IDATE = 33                   ' short date format, 0 M D Y, 1 D M Yr, 2 Y M D
CONSTANT long LOCALE_ILDATE = 34                  ' long date format, 0 M D Y, 1 D M Y, 2 Y M D
CONSTANT long LOCALE_ITIME = 35                  ' time format, 0 AM/PM 12-hr format, 1 24-hr format
CONSTANT long LOCALE_ICENTURY = 36                ' Use full 4-digit century, 0 Two digit. 1 Full century
CONSTANT long LOCALE_ITLZERO = 37                 ' leading zeros in time field, 0 No, 1 yes
CONSTANT long LOCALE_IDAYLZERO = 38               ' leading zeros in day field, 0 No, 1 yes
CONSTANT long LOCALE_IMONLZERO = 39               ' leading zeros in month field, 0 No, 1 yes
CONSTANT long LOCALE_S1159 = 40                  ' AM designator
CONSTANT long LOCALE_S2359 = 41                  ' PM designator
CONSTANT long LOCALE_SDAYNAME1 = 42               ' long name for Monday
CONSTANT long LOCALE_SDAYNAME2 = 43               ' long name for Tuesday
CONSTANT long LOCALE_SDAYNAME3 = 44               ' long name for Wednesday
CONSTANT long LOCALE_SDAYNAME4 = 45               ' long name for Thursday
CONSTANT long LOCALE_SDAYNAME5 = 46               ' long name for Friday
CONSTANT long LOCALE_SDAYNAME6 = 47               ' long name for Saturday
CONSTANT long LOCALE_SDAYNAME7 = 48               ' long name for Sunday
CONSTANT long LOCALE_SABBREVDAYNAME1 = 49          ' abbreviated name for Monday
CONSTANT long LOCALE_SABBREVDAYNAME2 = 51          ' abbreviated name for Tuesday
CONSTANT long LOCALE_SABBREVDAYNAME3 = 52          ' abbreviated name for Wednesday
CONSTANT long LOCALE_SABBREVDAYNAME4 = 53          ' abbreviated name for Thursday
CONSTANT long LOCALE_SABBREVDAYNAME5 = 54          ' abbreviated name for Friday
CONSTANT long LOCALE_SABBREVDAYNAME6 = 55          ' abbreviated name for Saturday
CONSTANT long LOCALE_SABBREVDAYNAME7 = 56          ' abbreviated name for Sunday
CONSTANT long LOCALE_SMONTHNAME1 = 57              ' long name for January
CONSTANT long LOCALE_SMONTHNAME2 = 58              ' long name for February
CONSTANT long LOCALE_SMONTHNAME3 = 59              ' long name for March
CONSTANT long LOCALE_SMONTHNAME4 = 60              ' long name for April
CONSTANT long LOCALE_SMONTHNAME5 = 61              ' long name for May

```

```

CONSTANT long LOCALE_SMONTHNAME6 = 62      ' long name for June
CONSTANT long LOCALE_SMONTHNAME7 = 63      ' long name for July
CONSTANT long LOCALE_SMONTHNAME8 = 64      ' long name for August
CONSTANT long LOCALE_SMONTHNAME9 = 65      ' long name for September
CONSTANT long LOCALE_SMONTHNAME10 = 66     ' long name for October
CONSTANT long LOCALE_SMONTHNAME11 = 67     ' long name for November
CONSTANT long LOCALE_SMONTHNAME12 = 68     ' long name for December
CONSTANT long LOCALE_SABBREVMONTHNAME1 = 69 ' abbreviated name for January
CONSTANT long LOCALE_SABBREVMONTHNAME2 = 70 ' abbreviated name for February
CONSTANT long LOCALE_SABBREVMONTHNAME3 = 71 ' abbreviated name for March
CONSTANT long LOCALE_SABBREVMONTHNAME4 = 72 ' abbreviated name for April
CONSTANT long LOCALE_SABBREVMONTHNAME5 = 73 ' abbreviated name for May
CONSTANT long LOCALE_SABBREVMONTHNAME6 = 74 ' abbreviated name for June
CONSTANT long LOCALE_SABBREVMONTHNAME7 = 75 ' abbreviated name for July
CONSTANT long LOCALE_SABBREVMONTHNAME8 = 76 ' abbreviated name for August
CONSTANT long LOCALE_SABBREVMONTHNAME9 = 77 ' abbreviated name for September
CONSTANT long LOCALE_SABBREVMONTHNAME10 = 78 ' abbreviated name for October
CONSTANT long LOCALE_SABBREVMONTHNAME11 = 79 ' abbreviated name for November
CONSTANT long LOCALE_SABBREVMONTHNAME12 = 80 ' abbreviated name for December
CONSTANT long LOCALE_SABBREVMONTHNAME13 = 4111 ' Native abbreviated name for 13th month, if i

string ls_buffer
ulong lul_size

string ls_str
ls_str = space(128)
GetLocaleInfoA( GetSystemDefaultLCID(), LOCALE_SNATIVELANGNAME, ls_str, 128)
MessageBox("", ls_str)

```

NOTE: Some values are not available on older Windows versions. You may want to verify on [Microsoft MSDN](#).

You can launch the Windows *Regional Settings Applets* with the following line:

```
Run( "rundll32.exe shell32.dll,Control_RunDLL intl.cpl,,4")
```

You may want to check this [HowTo](#).

## Make a window unmoveable

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0197.html>

Map pbm\_nclbuttondown to your own user event, then from your user event

```

IF hittestcode = 2 THEN // HTCAPTION
    message.processed = TRUE
    RETURN 1
END IF

```

```
RETURN 0
```

## Retrieve window handle by its title

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0199.html>

```
[local fucntion declaration]
FUNCTION ulong FindWindowA(ulong classname, string windowname) &
LIBRARY "user32.dll"

[powerscript]
public function unsignedlong uf_findwindow (string as_name);
//
// as_name:    Name of window (case sensitive)
//
// Returns: Window handle or zero if not found
//

ulong      ul_class

SetNull(ul_class)
RETURN FindWindowA(ul_class,as_name)
```

## Have a transparent window

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0210.html>

[Available on W2K or better] A cool effect giving a see-through window.

```
[local external function]
FUNCTION long GetWindowLong (ulong hWnd, int nIndex) &
LIBRARY "USER32.DLL" ALIAS FOR "GetWindowLongA"
FUNCTION long SetWindowLong (ulong hWnd, int nIndex, long dwNewLong) &
LIBRARY "USER32.DLL" ALIAS FOR "SetWindowLongA"

//W2K or better
FUNCTION long SetLayeredWindowAttributes &
(long hWnd, Long crKey , char /*Byte*/ bAlpha , Long dwFlags) &
LIBRARY "USER32.DLL"

[powerscript]
```

```
CONSTANT long LWA_COLORKEY = 1, LWA_ALPHA = 2
CONSTANT long GWL_EXSTYLE = - 20
CONSTANT long WS_EX_LAYERED = 524288 //2^19
long ll_Ret, ll_handle

// or-bitwise function
OleObject wsh
wsh = CREATE OleObject
wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"

ll_handle = Handle (this) // handle of the window
ll_Ret = GetWindowLong(ll_handle, GWL_EXSTYLE)
ll_Ret = wsh.Eval(string(ll_ret) + " or " + string(WS_EX_LAYERED))
SetWindowLong (ll_handle, GWL_EXSTYLE, ll_Ret)

// Set the opacity of the layered window to 128 (transparent)
SetLayeredWindowAttributes (ll_handle, 0, char(128),LWA_ALPHA)

// Set the opacity of the layered window to 255 (opaque)
// SetLayeredWindowAttributes (ll_handle, 0, char(255),LWA_ALPHA)
```

## Bypass Window Error popup message

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0212.html>

```
[local external function]
FUNCTION ulong SetErrorMode(ulong uMode) LIBRARY "KERNEL32.DLL"
```

The possible parameter values are:

```
CONSTANT ulong SEM_FAILCRITICALERRORS 1
CONSTANT ulong SEM_NOGPFAULTERRORBOX 2
CONSTANT ulong SEM_NOALIGNMENTFAULTEXCEPT 4
CONSTANT ulong SEM_NOOPENFILEERRORBOX 32768
```

Note: To combine values you can simply add them together.

## Get hard disk serial number

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0214.html>

```
[local external function declaration]
FUNCTION long GetVolumeInformation &
(string lpRootPathName, REF string lpVolumeNameBuffer, &
long nVolumeNameSize, &
REF long lpVolumeSerialNumber, REF long lpMaximumComponentLength, &
REF long lpFileSystemFlags, REF string lpFileSystemNameBuffer, &
long nFileSystemNameSize) &
LIBRARY "Kernel32.dll" ALIAS FOR "GetVolumeInformationA"

[powerscript]
String ls_volbuffer, ls_fsname
Long ll_serial, ll_MaxCompLength, ll_FileSystemFlags, ll_rtn

ls_volbuffer = Space(255)
ls_fsname = Space(255)
ll_maxCompLength = 0
ll_FileSystemFlags = 0

ll_rtn = GetVolumeinformation("C:\", ls_volbuffer, 255, ll_serial, &
                           ll_MaxCompLength, ll_FileSystemFlags , ls_fsname, 255)

// ls_volbuffer - volume name
// ll_serial      - hard disk serial number
// ls_fsname       - file system name ex. NTFS
```

## Make a selected item in a ListView visible

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0124.html>

```
int li_index
listviewitem llvi_data
string ls_name
CONSTANT int LVM_FIRST = 4096
CONSTANT int LVM_ENSUREVISIBLE = LVM_FIRST + 19

ls_name = "howto"

// search for "howto" item and select it
li_index = lv_1.FindItem(0,ls_name, FALSE, FALSE)
IF li_index > 0 THEN
  lv_1.GetItem(li_index, llvi_data)
  llvi_data.HasFocus = TRUE
  llvi_data.Selected = TRUE
  lv_1.SetItem(li_index,llvi_data)
END IF
```

```
Send(Handle(lv_1), LVM_ENSUREVISIBLE , li_index, 0)
```

## Get the CPU speed

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0221.html>

With NT (or better), the cpu speed is stored in the registry at

```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor\0  
~Mhz (dword)
```

The value is the Mhz - 1 , so for 2.4 Ghz, the value is 2399.

```
ulong speed  
string cpuname  
  
RegistryGet( &  
"HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor\0", &  
"~~MHz", ReguLong!, speed)  
  
RegistryGet( &  
"HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor\0", &  
"ProcessorNameString", RegString!, cpuname)  
  
MessageBox(cpuname, string(speed) + "Mhz")
```

## Disable the logon and logoff button

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0226.html>

1. Locate the registry entry  
HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer.
2. Right-click the Explorer key, and select New | DWORD value.
3. Name the value StartMenuLogoff
4. Type 1 in the Value Data text box
5. Right-click the Explorer key, and select New | DWORD value.
6. Name the value NoClose
7. Type 1 in the Value Data text box

8. Close the Registry Editor.
9. Reboot

## Disable the screensaver

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0228.html>

### XP

1. Locate the registry entry HKEY\_CURRENT\_USER\Control Panel\Desktop.
2. Locate the value ScreenSaveActive
3. Type 1 in the Value Data text box
4. Close the Registry Editor.

## List Windows processes or services

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0229.html>

Something similar to the Task List.

```
OleObject mssc
int li_rc
string ls_code
Any res

mssc = CREATE OleObject
li_rc = mssc.ConnectToNewObject( "MSScriptControl.ScriptControl" )
mssc.language = "VBScript"

ls_code = "function services() ~r~n" + &
          "strComputer=~".~"~r~n" + &
          "Set objWMIService = GetObject(~"winmgmts:~"" + &
          "& ~"{impersonationLevel=impersonate}!\"~" + &
          " & strComputer & ~"\root\cimv2~")~r~n" + &
          "Set colProcesses = objWMIService.ExecQuery(" + &
          "~"~select * from win32_process~" )~r~n" + &
          "s = ~"~"~r~n" + &
          "For Each objProcess In colProcesses~r~n" + &
          " s = s & objProcess.Name & vbCr~r~n" + &
          "Next~r~n" + &
          "services = s~r~n" + &
          "end function"
mssc.AddCode(ls_code)
res = mssc.Eval("services()")
MessageBox("", String(res))
```

```
mssc.DisconnectObject()
DESTROY mssc
```

You can get the list of the running services by replacing "select \* from win32\_process" by "select \* from win32\_service"

Useful links

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_process.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_process.asp)  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32\\_service.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/win32_service.asp)

## Terminate a Windows process

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0230.html>

In this HowTo, we are killing all the Notepad instance currently running, the script returns the number of instance found.

```
OleObject mssc
int li_rc
string ls_code
long i

mssc = CREATE OleObject
li_rc = mssc.ConnectToNewObject( "MSScriptControl.ScriptControl" )
mssc.language = "VBScript"

ls_code = "function killnotepad() ~r~n" + &
"Set locator = CreateObject(~"WbemScripting.SWbemLocator~")~r~n" + &
"Set service = locator.ConnectServer()~r~n" + &
"Set props = service.ExecQuery" + &
"(~"select name, description from Win32_Process" + &
" where name = 'notepad.exe'~")~r~n" + &
"num = props.count~r~n" + &
"for each notepad in props~r~n " + &
"    notepad.terminate ~r~n " + &
"next~r~n" + &
"killnotepad = num~r~n" + &
"end function"
// messagebox("", ls_code)
mssc.AddCode(ls_code)
i = mssc.Eval("killnotepad()")

messagebox("Notepad killed", string(i))

mssc.DisconnectObject()
DESTROY mssc
```

You need to run this code from an executable. From the IDE, the Terminate method returns "2" (Access is denied).

## Terminate a Windows application

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0266.html>

### Using VBScript

This will immediately close any running instance of Notepad.

```
OleObject wsh
integer li_rc

wsh = CREATE OleObject
wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"
wsh.AddCode('function terminatenotepad() ~n ' + &
'strComputer = "." ~n ' + &
'Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2") ~n ' + &
''Set colItems = objWMIService.ExecQuery("Select * from Win32_Process where name = ~'notepad~n')
'For Each objItem in colItems ~n ' + &
'    objItem.Terminate ~n ' + &
'Next ~n ' + &
'end function ~n ' )
wsh.executestatement('terminatenotepad()')
wsh.DisconnectObject()
DESTROY wsh
```

Replace notepad.exe with the executable name that you want to close.

### Using WinAPI with the class name

This will close a running Notepad but will ask to save any unsaved document. We are using the WinAPI to retrieve an handle and then we send the event WM\_CLOSE to the object associated with the handle.

```
[local external function declaration]
// PB10
FUNCTION ulong FindWindow(ref string classname, ref string windowname) &
LIBRARY "user32.dll" &
ALIAS FOR "FindWindowA;ansi"

// or PB10 alternate declaration
// FUNCTION ulong FindWindow (ref string classname, ref string windowname) &
// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowW"
```

```

// or PB9
// FUNCTION ulong FindWindow(ref string classname, ref string windowname) &
// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowA"

[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_class = "Notepad"
SetNull(ls_name)

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10 == 16
    send(hwnd, 16, 0, 0)
END IF

```

You need a **special tool** to detect the class name. A well known tool to detect the class name is SPY++, you may already have it (since it was included with PB Entreprise). Or you can download a free one, WinID at <http://www.softpedia.com/get/Programming/Debuggers-Decompilers-Dissassemblers/WinID.shtml>

To close an application builded with Powerbuilder, you need a class name, it's difficult because the class name is not always the same with new PB release. With PB10, it's FNWND3100 so for PB11 it maybe FNWND3110 ...

PB9	ls_class = "FNWND390"
PB10	ls_class = "FNWND3100"
PB11	ls_class = "FNWND3110" // ??

You may need to use the next technique if you want to close another Powerbuilder application and not close yourself at the same time !

#### Using WinAPI with a window title

Another possibility is to use the window title with the FindWindow API to search for the handle. This is ok if your title is static and known in advance.

```

[local external function declaration]
// PB10
FUNCTION ulong FindWindow(ref string classname, ref string windowname) LIBRARY "user32.dll"
ALIAS FOR "FindWindowA;ansi"

// or PB10 alternate declaration
// FUNCTION ulong FindWindow (ref string classname, ref string windowname) &
// LIBRARY "user32.dll" &
// ALIAS FOR "FindWindowW"

```

```
// or PB9
// FUNCTION ulong FindWindow(ref string classname, ref string windowname)  &
// LIBRARY "user32.dll"  &
// ALIAS FOR "FindWindowA"

[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_name = "Untitled - Notepad"
SetNull(ls_class)

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10
    send(hwnd, 16, 0, 0)
END IF
```

Note : It's possible to search the handle with the class name and the window title.

```
[powerscript]

ulong hWnd
string ls_title
string ls_class

ls_name = "Untitled - Notepad"
ls_class= "Notepad"

hWnd = FindWindow(ls_class, ls_title)

IF NOT IsNull(hWnd) THEN
    // WM_CLOSE = &H10
    send(hwnd, 16, 0, 0)
END IF
```

## Control CD audio tray, play MP3 file

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0232.html>

```
[external function declaration]
FUNCTION ULong MciSendString(String lpszCommand,  &
                           ref String lpszReturnString, &
```

```

    ULong cchReturn, ULong hwndCallback) &

LIBRARY "winmm.dll" &

ALIAS FOR "mciSendStringA"

[openCD function]
String lpszReturnString
ULong cchReturn, ll_rc

lpszReturnString = Space(100)
ll_rc = MciSendString &
('set cdaudio door open wait', lpszReturnString, cchReturn, 0)
// ll_rc = 0 Ok

[closeCD function]
String lpszReturnString
ULong cchReturn, ll_rc

lpszReturnString = Space(100)

ll_rc = MciSendString &
('set cdaudio door closed wait', lpszReturnString, cchReturn, 0)
//ll_rc = 0 Ok

```

MCI (Media Control Interface) provides the ability to play and record (as appropriate) on any supported multimedia device. To start the music from CD, you need to send the string "*play audio*" and "*close cdaudio*" to stop the music.

See <http://www.compguy.com/cwtip05.htm> for a command list.

To play MP3 (remember to declare the MciSendString external function!)

```

String lpszReturnString
string ls_mp3 = "C:\MP3\25. David Bowie - Heroes.mp3"
ULong cchReturn, ll_rc

lpszReturnString = Space(100)
ll_rc = MciSendString &
('close MP3_Device', lpszReturnString, cchReturn, 0)
ll_rc = MciSendString &
("open ~" + ls_mp3 + "~ type MPEGVideo alias MP3_Device", &
lpszReturnString, cchReturn, 0)
ll_rc = MciSendString &
("play Mp3_Device", lpszReturnString, cchReturn, 0)
// ll_rc = 0 Ok!

```

and to stop

```

ll_rc = MciSendString &
('close MP3_Device', lpszReturnString, cchReturn, 0)

```

If you have multiple CD drives, you can define alias for each one of them.

```
mciSendString("open E:\\ type cdaudio alias cdaudiol", 0, 0, 0);
mciSendString("open F:\\ type cdaudio alias cdaudio2", 0, 0, 0);
```

(MSDN) Multimedia Command Strings : <http://msdn2.microsoft.com/en-us/library/ms712587.aspx>

For a VBScript/Powershell solution, see this [HowTo](#)

## Detect if user is using Terminal Server

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0233.html>

```
[external function declaration]
FUNCTION Integer GetSystemMetrics (Integer nIndex) LIBRARY "user32.dll"

[powerscript]
integer SM_REMOTESESSION = 4096 // remote session
integer SM_REMOTECONTROL = 8193 // remote control

IF GetSystemMetrics(SM_REMOTESESSION)> 0 THEN
    // on terminal server
END IF
```

Another way is to check the Windows environment variable *sessionname*. The value of this environment variable will be 'Console' for a normal, local session. For an Remote Desktop session it will contain the phrase 'RDP'.

```
ContextKeyword lcxk_base
string ls_Path
string ls_values[]

this.GetContextService("Keyword", lcxk_base)
lcxk_base.GetContextKeywords("sessionname", ls_values)

MessageBox("", ls_values[1])
```

## Detect if a network is present

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0234.html>

```
[external function declaration]
FUNCTION Integer GetSystemMetrics (Integer nIndex) LIBRARY "user32.dll"

[powerscript]
integer SM_NETWORK = 63

IF GetSystemMetrics(SM_NETWORK) > 0 THEN
    MessageBox("", "Network is up")
END IF
```

See this related [HowTo](#)

## Detect network and/or internet connectivity state

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0241.html>

```
[Local function declaration]
FUNCTION boolean IsNetworkAlive(ref int flags) LIBRARY "sensapi.dll"

[Powerscript]
// The computer has one or more LAN cards that are active.
CONSTANT integer NETWORK_ALIVE_LAN = 1;
// The computer has one or more active RAS connections (internet).
CONSTANT integer NETWORK_ALIVE_WAN = 2;
// Win9x. The computer is connected to the America Online network.
CONSTANT integer NETWORK_ALIVE_AOL = 4;

OleObject wsh
int networkState, k

// to do the "bitwise AND" later...
wsh = CREATE OleObject
wsh.ConnectToNewObject( "MSScriptControl.ScriptControl" )
wsh.language = "vbscript"

IF IsNetworkAlive(networkState) THEN
    // check if a network card is active
    k = integer(wsh.Eval( string(networkState) + &
                           " AND " + &
                           string(NETWORK_ALIVE_LAN) ))
    IF k = NETWORK_ALIVE_LAN THEN
        MessageBox("", "network is ON")
    ELSE
        MessageBox("", "network is OFF")
    END IF
```

```
// check if internet connection is active
k = integer(wsh.Eval( string(networkState) + &
                      " AND " + &
                      string(NETWORK_ALIVE_WAN) ))
IF k = NETWORK_ALIVE_WAN THEN
    MessageBox("", "internet is ON")
ELSE
    MessageBox("", "internet is OFF")
END IF
ELSE
    MessageBox("", "no network ?!?")
END IF
```

See this related [HowTo](#)

## Detect Caps Lock state

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0236.html>

```
[local function declaration]
FUNCTION int GetKeyState(int keystatus) LIBRARY "user32.dll"

[powerscript]
int li_keystate

li_keystate = GetKeyState(20)

IF li_keystate = 1 THEN
    MessageBox("", "CAPS on")
ELSEIF li_keystate = 0 THEN
    MessageBox("", "CAPS off")
END IF
```

See also this [HowTo](#)

## Set or Unset the keyboard numlock and read its state

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/powerbuilder-set-or-unset-keyboard-numlock.html>

This code will toggle the keyboard NUM LOCK state.

```
OleObject wsh
```

```

Integer li_rc

wsh = CREATE OleObject
li_rc = wsh.ConnectToNewObject( "WScript.Shell" )

wsh.SendKeys ("{NUMLOCK}")

```

See this [HowTo](#) for a list of special keys that can be used (like CAPSLOCK).

To read the NUM LOCK state

```

[local function declaration]
FUNCTION int GetKeyState(int keystatus) LIBRARY "user32.dll"

[powerscript]

int li_keystate

li_keystate = GetKeyState(144)

IF li_keystate = 1 THEN
    MessageBox("", "NUMLOCK on")
ELSEIF li_keystate = 0 THEN
    MessageBox("", "NUMLOCK off")
END IF

```

## Convert Win API calls to PB10

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0237.html>

Function declaration needs to be modified to specify ANSI:

```

FUNCTION long ShellExecuteA (ulong hWnd, &
                           string Operation, string lpFile, string lpParameters, &
                           string lpDirectory, long nShowCmd) LIBRARY "shell32.dll" &
                           ALIAS FOR "ShellExecuteA;Ansi"

```

From the PowerBuilder 10 Help:

If your application makes calls to external functions, and there is a Unicode version of the external function, convert the calls to the Unicode versions of the functions. For example, a call to SQLGetInfo should be changed to SQLGetInfoW.

If the function passes a string, char, or structure as an argument or returns a string, char, or structure, you can use the same syntax in PowerBuilder 10 as in previous releases if the string uses Unicode encoding. For

example:

```
FUNCTION int MyMessageBoxW(int handle, string content, string title, int
showtype) LIBRARY "user32.dll" ALIAS FOR "MessageBoxW"
```

---

As for structure with PB10, char is now 16 bit :

```
// before PB10
global type mystruct from structure
    long l1
    char pad1
    char pad2
    long l2
    char pad3
    char pad4
end type

// PB10
global type mystruct from structure
    long l1
    char pad1
    long l2
    char pad2
end type
```

## Hide a window from the Windows task list

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0239.html>

You can set the visible property of a window to false

```
w_window.visible = false
```

but the window is not visible anymore.

To keep the window visible but not in the task bar, define the external function

```
FUNCTION ulong SetWindowLongA( ulong hWnd, int nIndex, long newValue ) &
LIBRARY "USER32.DLL"
```

and then in the window open event

```
long ll_rc

ll_rc = SetWindowLongA(Handle(This), -20, 128)
IF ll_rc = 0 THEN
```

```
// something wrong on the kingdom!
END IF
```

## Get the number of items in a Treeview

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0242.html>

```
CONSTANT integer TV_FIRST = 4352
CONSTANT integer TVM_GETCOUNT = TV_FIRST + 5

long ll_tvitemcount
ll_tvitemcount = Send ( handle(tv_1), TVM_GETCOUNT, 0, 0 )
```

## Detect if the current user is a member of the Administrator's group

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0245.html>

[W2K/XP ok but not Vista!]

```
[local function declaration]
FUNCTION boolean IsUserAnAdmin() LIBRARY "shell32.dll"

[powerscript]
MessageBox("Is Admin ?", IsUserAnAdmin())
```

With Vista, you need to use the LookupPrivilegeValue and AdjustTokenPrivileges APIs, I have no example but if you have one, send it to me!

## Resize a window because XP window title is bigger

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0247.html>

XP theme, when in used, makes the title window bigger. But if the window design was done with a traditional Windows, you may lose information in the windows bottom area.

This snippet will detect if the title is bigger than the traditional one. If it's bigger then the window is resized.

```
[local function declaration]
```

```
FUNCTION integer GetSystemMetrics ( int nIndex ) LIBRARY "user32.dll"

[window open event]
CONSTANT integer SM_CYCAPTION = 4
integer li_pixels, li_pbunits

li_pixels = GetSystemMetrics(SM_CYCAPTION)
li_pbunits = PixelsToUnits(li_pixels, YPixelsToUnits!)

this.height = this.height + ( li_pbunits - 76 )
```

## Convert the datatypes from Microsoft Win API to PowerBuilder

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0248.html>

MICROSOFT	PB(32Bit)
Bool	Boolean
Char*	Ref String
Colorref	Ulong
Dword	Ulong
Handle	Ulong
Hdc	Ulong
Hfile	Ulong
Hinstance	Ulong
Hwnd	Ulong
Int	Int
Lparam	Ulong
Lpbyte	Ref Long
Lpdword	Ref Ulong
Lpfiletime	Ref Time
Lpint	Ref Long
Lpstr, Lpststr	Ref String
Lpvoid	Ref Struct struct_inst
Mcierror	Long
Lpstr, Lpststr	Ref String
Lpvoid	Ref Struct struct_inst
Pbyte	Ref Long[#]
Short	Int
Structure	Ref Struct Struct_inst
Uint	Uint
Void**	SUBROUTINE
Word	Ulong
Wparam	Ulong

## Allow user:password in URL

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0249.html>

The following URL syntax is no longer supported in Internet Explorer or in Windows Explorer after you install the MS04-004 Cumulative Security Update for Internet Explorer (832894):

```
http(s)://username:password@server/resource.ext
```

This change in the default behavior is also implemented by security updates and service packs that were released after the 832894 security update.

By default, this new default behavior for handling user information in HTTP or HTTPS URLs applies only to Windows Explorer and Internet Explorer. To use this new behavior in other programs that host the Web browser control, create a DWORD value named SampleApp.exe, where SampleApp.exe is the name of the executable file that runs the program. Set the DWORD value's value data to 1 in one of the following registry keys.

- For all users of the program, set the value in the following registry key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Internet Explorer\
    Main\FeatureControl\FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
```

- For the current user of the program only, set the value in the following registry key:

```
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\
    Main\FeatureControl\FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
```

To disable the new default behavior in Windows Explorer and Internet Explorer, create iexplore.exe and explorer.exe DWORD values in one of the following registry keys and set their value data to 0.

- For all users of the program, set the value in the following registry key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Internet Explorer\
    Main\FeatureControl\FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
```

- For the current user of the program only, set the value in the following registry key:

```
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\
    Main\FeatureControl\FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
```

ref [Microsoft Article ID : 834489](#)

## Get the time and date from a server

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0250.html>

First we need these local function declarations

```
[local function declaration]
FUNCTION ULONG GetTimeOfDayInfo &

    ( REF s_timeofday dest, ULONG source, ULONG size ) &
    LIBRARY "KERNEL32.DLL" ALIAS FOR "RtlMoveMemory"

FUNCTION ULONG NetRemoteTOD &
```

```
(REF CHAR server[], REF ULONG bufferPtr) LIBRARY "NETAPI32.DLL"
FUNCTION ULONG LocalhostTOD &
    (ref long null, REF ULONG bufferPtr) LIBRARY "NETAPI32.DLL" &
        ALIAS FOR "NetRemoteTOD"
```

this structure

```
[s_timeofday structure]
elapsedt uLong
msecs uLong
hours uLong
mins uLong
secs uLong
hunds uLong
timezone Long
tinterval uLong
day uLong
month uLong
year uLong
weekday uLong
```

and these functions

```
[powerscript]

// -----
of_StringToUnicode (String as_string, ref character ac_unicode[])

int li_loop, li_len, li_uni

li_len = Len( as_string )
FOR li_loop = 1 TO li_len
    li_uni++
    ac_unicode[ li_uni ] = Mid( as_string, li_loop, 1 )
    li_uni++
    ac_unicode[ li_uni ] = Char( 0 )
NEXT

li_uni++
ac_unicode[ li_uni ] = Char( 0 )
li_uni++
ac_unicode[ li_uni ] = Char( 0 )

// -----
datetime of_relativetimedate(datetime adt_start, long al_offset)

// stolen from the PFC!
datetime ldt_null
date ld_sdate
time lt_stime
long ll_date_adjust
```

```

long ll_time_adjust, ll_time_test

//Initialize date and time portion
ld_sdate = date(adt_start)
lt_stime = time(adt_start)

//Find out how many days are contained
//Note: 86400 is # of seconds in a day
ll_date_adjust = al_offset / 86400
ll_time_adjust = mod(al_offset, 86400)

//Adjust date portion
ld_sdate = RelativeDate(ld_sdate, ll_date_adjust)

//Adjust time portion
// Allow for time adjustments periods crossing over days
// Check for time rolling forwards a day
IF ll_time_adjust > 0 THEN
    ll_time_test = SecondsAfter(lt_stime, time('23:59:59'))
    IF ll_time_test < ll_time_adjust THEN
        ld_sdate = RelativeDate(ld_sdate, 1)
        ll_time_adjust = ll_time_adjust - ll_time_test -1
        lt_stime = time('00:00:00')
    END IF
    lt_stime = RelativeTime(lt_stime, ll_time_adjust)
//Check for time rolling backwards a day
ELSEIF ll_time_adjust < 0 THEN
    ll_time_test = SecondsAfter(lt_stime, time('00:00:00'))
    IF ll_time_test > ll_time_adjust THEN
        ld_sdate = RelativeDate(ld_sdate, -1)
        ll_time_adjust = ll_time_adjust - ll_time_test +1
        lt_stime = time('23:59:59')
    END IF
    lt_stime = RelativeTime(lt_stime, ll_time_adjust)
END IF

RETURN (datetime(ld_sdate, lt_stime))

```

then the actual call to get the time and date from a server (or the localhost), we need to adjust the value returned because of the timezone offset.

```

[powerscript]

char lc_logonserver[]
string ls_logonserver
boolean lb_logonserver

ulong ul_prtTimeOfDayInfo
ulong ul_rc
long ll_null

s_timeofday lstr_timedate
ContextKeyword lcxk_base
String ls_result

```

```

string ls_values[]

datetime ldt_locale
date ld_server
time lt_server
datetime ldt_server

// get the logon server
this.GetContextService("Keyword", lcwk_base)
lcwk_base.GetContextKeywords("LOGONSERVER", ls_values)
IF Upperbound(ls_values) > 0 THEN
    ls_logonserver = ls_values[1]
    // transform logon server to unicode
    of_StringToUnicode(ls_logonserver, lc_logonserver)
    lb_logonserver = true
ELSE
    // lc_logonserver is null --> no server -> localhost
    lb_logonserver = false
END IF

// get the current time of day
IF lb_logonserver THEN
    ul_rc = NetRemoteTOD(lc_logonserver, ul_prtTimeOfDayInfo)
ELSE
    ll_null = 0
    ul_rc = LocalhostTOD(ll_null, ul_prtTimeOfDayInfo)
END IF
//

IF NOT ul_rc = 0 THEN
    MessageBox("Oups", String (ul_rc))
ELSE
    // convert to our structure
    GetTimeOfDayInfo(lstr_timedate, ul_prtTimeOfDayInfo, 48)
    ls_result = string(lstr_timedate.year) + "-" +
        + string(lstr_timedate.month) +
        + "-" + string(lstr_timedate.day) +
        + " " + string(lstr_timedate.hours) + ":" +
        + string(lstr_timedate.mins) +
        + ":" + string(lstr_timedate.secs)
    MessageBox("result before conversion" , ls_result )

    // convert to local time taking into account the timezone
    ld_server = Date (string(lstr_timedate.year) + "-" +
        + string(lstr_timedate.month) +
        + "-" + string(lstr_timedate.day))
    lt_server = Time (string(lstr_timedate.hours) + ":" +
        + string(lstr_timedate.minutes) +
        + ":" + string(lstr_timedate.seconds))

```

```

+ string(lstr_timedate.mins) + &
":" + string(lstr_timedate.secs))

ldt_locale = of_relativeDatetime &
(datetime(ld_server, lt_server), &
- (lstr_timedate.timezone * 60))
Messagebox("result after conversion", &
String(ldt_locale, "yyyy-mm-dd hh:mm"))
END IF

```

See also this [HowTo](#)

## Set the wallpaper

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0253.html>

First we need a local function declaration

```
[local function declaration]
FUNCTION ulong SystemParametersInfoA( ulong ul_action, &
ulong ul_iParam, string s_bitmap, ulong ul_winIni ) &
LIBRARY "USER32"
```

And then from Powerscript

```
[powerscript]
CONSTANT ulong SPISETDESKWALLPAPER = 20
// to make the change permanent
CONSTANT ulong SPIUPDATEINIFILE = 1
string ls_wallpaper
// only BMP is possible
// to use a JPG, you need to convert it to BMP first.
ls_wallpaper = "C:\WINDOWS\Zapotec.bmp"
SystemParametersInfoA( SPISETDESKWALLPAPER, 0, &
ls_wallpaper, SPIUPDATEINIFILE )
```

## Scan Window titles

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0255.html>

This snippet displays all Internet Explorer window titles which are opened.

```
[local function declaration]
FUNCTION LONG GetDesktopWindow() LIBRARY "user32"
FUNCTION LONG GetWindow( long hWnd, long uCmd ) LIBRARY "user32"
FUNCTION LONG GetClassNameA &
( long hWnd, Ref String lpClassName, long nMaxCount ) &

LIBRARY "user32"
FUNCTION LONG GetWindowTextA &
( long hWnd, Ref String lpString, long nMaxCount ) &

LIBRARY "user32"
FUNCTION BOOLEAN IsWindowVisible( long hWnd ) LIBRARY "user32"

[powerscript]
long ll_Desktop,ll_Child

string ls_ClassName, ls_WindowName, ls_ieopened

Constant long GW_HWNDFIRST = 0
Constant long GW_HWNDLAST = 1
Constant long GW_HWNDNEXT = 2
Constant long GW_HWNDFPREV = 3
Constant long GW_OWNER = 4
Constant long GW_CHILD = 5
Constant long GW_MAX = 5

Constant long MAX_WIDTH = 255

ll_Desktop = GetDesktopWindow()

ll_Child = GetWindow( ll_Desktop, GW_CHILD )

ls_ieopened = ""
DO WHILE (ll_Child > 0)
  ls_ClassName = Space(MAX_WIDTH)
  ls_WindowName = Space(MAX_WIDTH)
  // Get window classname
  GetClassNameA( ll_Child, ls_ClassName, MAX_WIDTH )
  // Get window text
  GetWindowTextA( ll_Child, ls_WindowName, MAX_WIDTH )

  IF ls_classname = "IEFrame" THEN
```

```

    ls_ieopened += ls_ClassName + "(" + ls_WindowName + ")"
END IF
l1_Child = GetWindow( l1_Child, GW_HWNDNEXT )
LOOP

Messagebox("", ls_ieopened)

```

## Get an UUID (Universally Unique Identifiers)

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/./pbdetails/pb-0256.html>

Thanks to O.L.Dansereau for the tip!

UUID means *Universally Unique Identifier*, see this [Wikipedia article](#) for more details.

```

[structure definition]
TYPE s_uuid FROM structure
  unsignedlong  data1
  unsignedinteger  data2
  unsignedinteger  data3
  unsignedinteger  data4[4]
END TYPE

[local function declaration]
FUNCTION long uuidCreate(ref s_uuid astr_uuid) LIBRARY "Rpcrt4.dll"  &
  ALIAS FOR "UuidCreate"

[powerscript function string of_hex(unsignedlong aul_decimal) ]
String ls_hex
Character lch_hex[0 TO 15] = &
{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', &
 'c', 'd', 'e', 'f'}

// Check parameters
IF IsNull(aul_decimal) THEN
  SetNull(ls_hex)
  RETURN ls_hex
END IF

DO
  ls_hex = lch_hex[Mod(aul_decimal, 16)] + ls_hex
  aul_decimal /= 16
LOOP UNTIL aul_decimal = 0

RETURN ls_hex

```

```
[powerscript]
long ll_rc
s_uuid lstr_uuid
string ls_guid = ""

constant long RPC_S_OK = 0
constant long RPC_S_UUID_LOCAL_ONLY = 1824
constant long RPC_S_UUID_NO_ADDRESS = 1739

ll_rc = uuidCreate(lstr_uuid)
// returns
//   RPC_S_OK - The call succeeded.
//   RPC_S_UUID_LOCAL_ONLY -
//     The UUID is guaranteed to be unique to this computer only.
//   RPC_S_UUID_NO_ADDRESS -
//     Cannot get Ethernet/token-ring hardware address for this computer.
IF ll_rc <> RPC_S_OK THEN
    setNull(ls_GUID)
    // MessageBox("", "uuid create not ok ?!?")
ELSE
    ls_GUID = right("00000000" + of_hex(lstr_uuid.data1), 8)
    ls_GUID += "-" + right("0000" + of_hex(lstr_uuid.data2), 4)
    ls_GUID += "-" + right("0000" + of_hex(lstr_uuid.data3), 4)
    ls_GUID += "-" + right("0000" + of_hex(lstr_uuid.data4[1]), 4)
    ls_GUID += "-" + right("0000" + of_hex(lstr_uuid.data4[2]), 4) +
        + right("0000" + of_hex(lstr_uuid.data4[3]), 4) +
        + right("0000" + of_hex(lstr_uuid.data4[4]), 4)
    ls_GUID = upper(ls_GUID)
    // MessageBox("", ls_guid)
    // output example : 00003B93-2641-477A-C99E-A2FFEBEB214A
END IF
```

## Encode/decode base64 string

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0258.html>

WinXP (or better), this HowTo is based the Microsoft Crypto API.

Base64 can be used to quickly encoded/decode sensitive information in the Registry or INI file. Note that Base64 encoding is NOT a secure algorithm but it hides informations from casual readers.

The string *realhowto* in Base64 is *cmVhbGhvdkRv*.

Inspired by this [winsock user object](#).

A nice tool to check your encoded strings is <http://paulschou.com/tools/xlate/>.

```

[local function declaration]
//BOOL WINAPI CryptBinaryToString(
//    const BYTE* pbBinary,
//    DWORD cbBinary,
//    DWORD dwFlags,
//    LPTSTR pszString,
//    DWORD* pcchString
//);
FUNCTION boolean CryptBinaryToString ( &
    Blob pbBinary, &
    ulong cbBinary, &
    ulong dwFlags, &
    Ref string pszString, &
    Ref ulong pcchString ) &
LIBRARY "crypt32.dll" ALIAS FOR "CryptBinaryToStringA"

//BOOL WINAPI CryptStringToBinary(
//    LPCTSTR pszString,
//    DWORD cchString,
//    DWORD dwFlags,
//    BYTE* pbBinary,
//    DWORD* pcbBinary,
//    DWORD* pdwSkip,
//    DWORD* pdwFlags
//);
// 

FUNCTION boolean CryptStringToBinary ( &
    string pszString, &
    ulong cchString, &
    ulong dwFlags, &
    Ref blob pbBinary, &
    Ref ulong pcbBinary, &
    Ref ulong pdwSkip, &
    Ref ulong pdwFlags ) &
LIBRARY "crypt32.dll" ALIAS FOR "CryptStringToBinaryA"

[instance variables]
// Base64, with certificate beginning and ending headers
CONSTANT Ulong CRYPT_STRING_BASE64HEADER = 0

// Base64, without headers
CONSTANT Ulong CRYPT_STRING_BASE64 = 1

// Pure binary copy
CONSTANT Ulong CRYPT_STRING_BINARY = 2

// Base64, with request beginning and ending headers
CONSTANT Ulong CRYPT_STRING_BASE64REQUESTHEADER = 3

// Hexadecimal only
CONSTANT Ulong CRYPT_STRING_HEX = 4

// Hexadecimal, with ASCII character display

```

```

CONSTANT Ulong CRYPT_STRING_HEXASCII = 5

// Base64, with X.509 CRL beginning and ending headers
CONSTANT Ulong CRYPT_STRING_BASE64X509CRLHEADER = 9

// Hexadecimal, with address display
CONSTANT Ulong CRYPT_STRING_HEXADDR = 10

// Hexadecimal, with ASCII character and address display
CONSTANT Ulong CRYPT_STRING_HEXASCIIADDR = 11

// A raw hex string. WinServer 2K3, WinXP: This value is not supported.
CONSTANT Ulong CRYPT_STRING_HEXRAW = 12

```

( [MSDN](#) )

#### Encode a String to Base64

```

String ls_encoded
ULong lul_len, lul_buflen
Boolean lb_rtn

Blob value

// the value to be encoded, needs to be 3 char or more (but not 4?)
value = blob(sle_1.text)

lul_len = Len(value)
lul_buflen = lul_len * 2
ls_encoded = Space(lul_buflen)

lb_rtn = CryptBinaryToString(value, &

                           lul_len, CRYPT_STRING_BASE64, &

                           ls_encoded, lul_buflen)

IF NOT lb_rtn THEN
  ls_encoded = ""
ELSE
  // remove the last two chr(0) !
  ls_encoded = left(ls_encoded, len(ls_encoded) - 2 )
END IF

sle_2.text = ls_encoded

```

#### Decode Base64 to a String

```

Blob lblob_data
ULong lul_len, lul_buflen, lul_skip, lul_pflags
Boolean lb_rtn

String value

```

```

value = sle_2.text // the value to be decoded

lul_len = Len(value)
lul buflen = lul_len
lblob_data = Blob(Space(lul_len))

lb_rtn = CryptStringToBinary(value, &

                           lul_len, CRYPT_STRING_BASE64, lblob_data, &

                           lul buflen, lul_skip, lul_pflags)

sle_1.text = string(BlobMid(lblob_data, 1, lul buflen))

```

---

Since PB10 or better handles string as Unicode (16-bits wide and not 8-bits like PB previous versions), you must change the function declarations to :

```

FUNCTION boolean CryptBinaryToString ( &

    Blob pbBinary, &

    ulong cbBinary, &

    ulong dwFlags, &

    Ref string pszString, &

    Ref ulong pcchString ) &

LIBRARY "crypt32.dll" ALIAS FOR "CryptBinaryToStringA;Ansi"
```

```

FUNCTION boolean CryptStringToBinary ( &

    string pszString, &

    ulong cchString, &

    ulong dwFlags, &

    Ref blob pbBinary, &

    Ref ulong pcbBinary, &

    Ref ulong pdwSkip, &

    Ref ulong pdwFlags ) &

LIBRARY "crypt32.dll" ALIAS FOR "CryptStringToBinaryA;Ansi"
```

When you encode the string, you need to make sure to store in blob an Ansi encoding string.

Encode a String to Base64 (PB10+)

```
...
value = blob(sle_1.text, EncodingANSI!)
...
```

## Get the executable name

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0265.html>

```
[local external function declaration]
FUNCTION ulong GetModuleFileName (ulong hinstModule, ref string lpszPath, ulong cchPath )  &
    LIBRARY "KERNEL32.DLL" &
    ALIAS FOR "GetModuleFileNameA;ansi" // ;ansi required for PB10 or better

[powerscript]
ClassDefinition lcd

String ls_fullpath
ulong lul_handle, lul_length = 512

IF handle(getapplication()) = 0 THEN
    // running from the IDE
    lcd=getapplication().classdefinition
    ls_fullpath = lcd.libraryname
ELSE
    // running from EXE
    lul_handle = handle( getapplication() )
    ls_fullpath=space(lul_length)
    GetModuleFilename( lul_handle, ls_fullpath, lul_length )
END IF

MessageBox("", ls_fullpath)
```

---

See also this [PB HowTo](#)

## Get Network card description and Mac address

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/pb-0267.html>

### Using VBScript

```
OLEObject ole_wsh
Any la_usb[]
string ls_message
```

```

ole_wsh = CREATE OLEObject
ole_wsh.ConnectToNewObject ("MSScriptControl.ScriptControl")
ole_wsh.Language = "vbscript"
ole_wsh.AddCode('Function rtnMACAddresses()~r~n' &
+ 'MACAddressList = "" ~r~n' + &
+ 'strComputer = "."~r~n' + &
+ 'Set objWMIService = ' + &
+ '   GetObject("winmgmts:{impersonationLevel=impersonate}!\\\" _~r~n' &
+ '& strComputer & "\\root\cimv2")~r~n' &
+ 'Set colItems = ' + &
+ '   objWMIService.ExecQuery("Select * from Win32_NetworkAdapterConfiguration Where IPEnabled = 1")~r~n' &
+ 'For Each objItem in colItems~r~n' &
+ 'MACAddressList = MACAddressList & " | " & objItem.Description & " = " & objItem.MACAddress &
+ 'Next~r~n' &
+ 'rtnMACAddresses = MACAddressList~r~n' &
+ 'End Function')
ls_message = ole_wsh.Eval("rtnMACAddresses")
ole_wsh.DisconnectObject()
DESTROY ole_wsh

MessageBox("MAC Adress List",ls_message)

```

## Turn on/off the monitor

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/powerbuilder-turn-on-off-the-monitor.html>

```

[local function declaration]
FUNCTION long SendMessageA &

(long hwnd,  long msg,  long wparam,  long lparam) &

LIBRARY "USER32.DLL"

[powerscript]
CONSTANT long SC_MONITORPOWER = 61808
CONSTANT long WM_SYSCOMMAND = 274
CONSTANT long MONITOR_ON = -1
CONSTANT long MONITOR_OFF = 2
// CONSTANT long MONITOR_STANDBY = 1

long hWnd
hWnd = handle(parent) // from a button, retrieve the window handle

SendMessageA(hWnd, WM_SYSCOMMAND, SC_MONITORPOWER, MONITOR_OFF)
Sleep(2)
SendMessageA(hWnd, WM_SYSCOMMAND, SC_MONITORPOWER, MONITOR_ON)

```

## Detect if running in 64bit OS

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/powerbuilder-detect-if-running-in-a-64bit-OS.html>

One way check is to check if the current Powerbuilder process is running in WOW64 mode (ie. the 32bit emulation mode).

[Local external definitions]

```
FUNCTION long IsWow64Process(long hwnd, ref boolean Wow64Process) &
LIBRARY "KERNEL32.DLL"

FUNCTION long GetCurrentProcess () LIBRARY "KERNEL32.DLL"
```

[Powerscript]

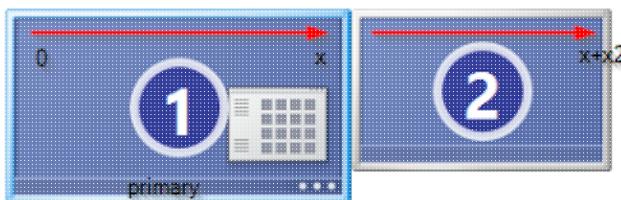
```
boolean wow64 =false
IsWow64Process(GetCurrentProcess(), wow64)
MessageBox("Running in 64b env", wow64)
```

## Display a window on the second monitor

Current version of this HowTo :

<https://www.rgagnon.com/pbdetails/..pbdetails/powerbuilder-display-on-the-second-monitor.html>

When you have more than one screen attached to a PC, one possible configuration is to see the dual monitors setup as a virtual desktop.



The trick is to get the resolution of the first monitor (the main one) and then position a window on the second screen (right monitor).

First you need to this API to detect if more than one monitor is present.

```
[local external function definition]
FUNCTION Integer GetSystemMetrics (Integer nIndex) LIBRARY "user32.dll"
```

In the open event of a window, we position the window at the max value of screen resolution returned by the Powerbuiler environment object.

```
integer SM_CMONITORS = 80
environment myEnv

IF GetSystemMetrics(SM_CMONITORS) > 1 THEN
    GetEnvironment(myEnv)
    // display on the secondary monitor (right to primary monitor!)
    this.x = PixelsToUnits(myEnv.ScreenWidth, XPixelsToUnits!)
    this.y = 0
END IF
```

Now if you want to center the window displayed on the second monitor, you need to get the resolution on that monitor. The Powerbuilder environment object returns only the resolution of the main monitor. To get the resolution of the second monitor, you need to call two Windows APIs.

```
[local external function definition]
FUNCTION Long MonitorFromWindow (Long hwnd, Long dwFlags) LIBRARY "user32"
FUNCTION Long GetMonitorInfo (long hMonitor, ref tagmonitorinfo moninfo) &
    LIBRARY "user32" ALIAS FOR "GetMonitorInfoA"
```

A structure, **tagmonitorinfo**, is defined to contain the informations returned by the API.

We position the window on the second monitor. We call the MonitorFromWindow API to get an identifier for the monitor as seen by Windows. Then with the identifier, we call another API to the associated screen resolution.

```
integer SM_CMONITORS = 80
integer MONITOR_DEFAULTTONULL = 0

IF GetSystemMetrics(SM_CMONITORS) > 1 THEN
    environment myEnv
    long monitor
    tagmonitorinfo tmi
    int x1
    int y1

    GetEnvironment(myEnv)
    // move to the second monitor (right to primary monitor!)
    this.x = PixelsToUnits(myEnv.ScreenWidth, XPixelsToUnits!)
    this.y = 0

    // get the resolution of the second monitor
    monitor = MonitorFromWindow(Handle(w_yop), MONITOR_DEFAULTTONULL)

    IF NOT isNull(monitor) THEN
        tmi.cbSize = 72 // size in bytes of TAGMONITORINFO
        GetMonitorInfo(monitor, tmi);
```

```

// the resolution
x1 = tmi.rcMonitor.right - tmi.rcMonitor.left;
y1 = tmi.rcMonitor.bottom - tmi.rcMonitor.top;

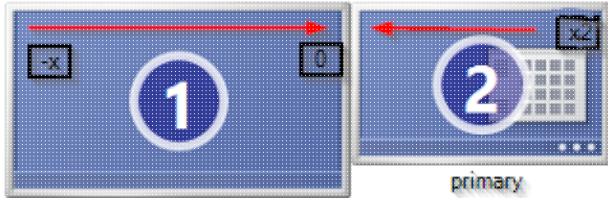
// display center on the secondary monitor (right to primary monitor!)
this.X = PixelsToUnits(myEnv.ScreenWidth, XPixelsToUnits!) + &

    (( PixelsToUnits(x1, XPixelsToUnits!) - this.Width ) / 2)
this.Y = ( PixelsToUnits( y1, YPixelsToUnits! ) - this.Height ) / 2

END IF
END IF

```

If the second monitor is to the left of the primary monitor then to position a window we need to use a negative value.



```

integer SM_CMONITORS  = 80
integer MONITOR_DEFAULTTONULL = 0

IF GetSystemMetrics(SM_CMONITORS)> 1 THEN
    environment myEnv
    long monitor
    tagmonitorinfo tmi
    int x1
    int y1

    GetEnvironment (myEnv)
    // move to the second monitor (left to primary monitor)
    this.x = -1 * (this.Width)
    this.y = 0

    // get the resolution of the second monitor
    monitor = MonitorFromWindow(Handle(w_yop), MONITOR_DEFAULTTONULL)

    IF NOTisNull(monitor) THEN
        tmi.cbSize = 72 // size in bytes of TAGMONITORINFO
        GetMonitorInfo(monitor, tmi);
        // the resolution
        x1 = tmi.rcMonitor.right - tmi.rcMonitor.left;
        y1 = tmi.rcMonitor.bottom - tmi.rcMonitor.top;

        // display center on the secondary monitor (left to primary monitor!)
        this.X = -1 * ( PixelsToUnits(x1, XPixelsToUnits!) - &

            (PixelsToUnits(x1, XPixelsToUnits!) - this.Width ) / 2)
        this.Y = ( PixelsToUnits( y1, YPixelsToUnits! ) - this.Height ) / 2
    END IF

```

END IF

---

---

Written and compiled Réal Gagnon ©2019 real@rgagnon.com  
<https://www.rgagnon.com>