

```
<?xml version="1.0" encoding="UTF-16"?>
<Module>
  <!-- This is a comment to help others read my script-->
  <!-- Global Script block. Constants, global variables and subroutines are placed here -->
  <?Script VBScript
    ...
  ?>
  <!-- Module properties -->
  <Description>This is a Script Module</Description>
  <RunWithShadow >true</RunWithShadow >
  <ShadowOverride>true</ShadowOverride>
  <Standalone>true</Standalone>
  <Events>
    <!-- Events section. Each event is called from the Palmbutler system in response to key events and other events-->
    <OnInitialization>
      <!-- This event is called right after the script is loaded. -->
    </OnInitialization>
    <OnFinalization>
      <!-- This event is called just before the script ends -->
    </OnFinalization>
    <OnKeyPlay >
      <!-- This event is called when the key "Play" is pressed on the Palmbutler remote -->
      <!-- An event consists of one or more actions. There are 3 types of actions: Action tag, Script and Script Call -->
      <!-- Example Action tag -->
      <RunProgram Filename="Notepad.exe" Arguments="c:\My Documents\abc.txt"/>
      <!-- Example Script -->
      <?Script VBScript
        set WshShell = CreateObject("WScript.Shell")
        WshShell.Run "calc"
        WshShell.AppActivate "Calculator"
      ?>
      <!-- Example Script Call. Calls the subroutine named "ClickCalculator" with one argument "15"-->
      <CallScript Name="ClickCalculator">
        <Argument>15</Argument>
      </CallScript>
    </OnKeyPlay >
  </Events>
  <!-- Global Script block. Global variables and subroutines are placed here -->
  <?Script VBScript
    Sub ClickCalculator(a1)
      set WshShell = CreateObject("WScript.Shell")
      WshShell.AppActivate "Calculator"
      WshShell.SendKeys a1
    End Sub
  ?>
</Module>
```

Figure 1: Outline of a Palmbutler Script Module (psm).

Key Events

All names of key events are made from the word “OnKey” and the name of the key. F.ex. “OnKeyStop” is the events that is triggered when the Stop key is pressed.

The Roadrunner terminal key names are listed here:

'Right',
'Down',
'Enter',
'Up',
'Left',
'VolUp',
'VolDown',
'Back',
'Mute',
'Menu',
'Prev',
'Play',
'Stop',
'Next',
'Red',
'Blue',
'Green',
'Shift',
'0',
'1',
'2',
'3',
'4',
'5',
'6',
'7',
'8',
'9',
'Star'

In addition to the simple key press events the following key events exists:

Shift key combination events

Double function key events

Repeat key events

Shift key event names are made of the word “OnKeyShift” and the name of the key. F.ex “OnKeyShiftStop” is the event that is triggered when the Shift and Stop key is pressed simultaneously.

Double function keys are used to put 2 different functions on the same key. F.ex a short key press on the Stop key will trigger the “OnKeyStop” event. A long key press on the Stop key (1 second) will trigger the “OnKeyHoldStop” event.

To make keys repeat use the word “OnKeyRepeat” and the name of the key. F.ex the “OnKeyRepeatStop” event is triggered repeatedly when you hold down the Stop key.

The Palmbutler scripting system automaticly configures the keys on the remote from the key events in the script file. It does this with the following priorities.

Priority	If the event exists	Key type	Active events
1	“OnKeyRepeat”	Repeating key	“OnKey” and “OnKeyRepeat”
2	“OnKeyHold”	Double function key	“OnKey” and “OnKeyHold”
3	“OnKeyShift”	Shift combination key	“OnKey” and “OnKeyShift”
4	“OnKey”	Single function key	“OnKey”

Therefore if f.ex the events “OnKeyStop” and “OnKeyHoldStop” and “OnKeyShiftStop” is defined in the script file the key will be configured as a double function key and the “OnKeyShiftStop” will never be triggered.

Some keys have a default type and function if not defined in the script file.

Key	Default function	Default key type
“Red”, “Green”, “Blue”	Hotkeys	Single
“VolUp”, “VolDown”	Volume control	Repeat
“Mute”	Volume mute	Single
“Back”	Back / Escape *)	Double *)
“Shift”	Shift key	Single *)

The keys or type marked with *) are permanent. The “Back” key is always a double function key and the second function is locked to the escape function. The first function of the “Back” can be changed with the “OnKeyBack” event. The shift key is always a single function key and can be programmed with the “OnKeyShift” event.

The Shift-Stop combination is reserved and can not be programmed in scripts

Actions

An event consists of one or more actions. There are 3 types of actions.

1. Script actions

A script action is a block of script commands. The script language can be Visual Basic Script or JavaScript or any other installed script engine. Here is an example of a Visual Basic Script block inside the “OnKeyEnter” event.

```
<OnKeyEnter>
  <?Script VBScript
    Set ThisWord = CreateObject("Word.Basic")
    With ThisWord
      .AppMaximize
      .FileNewDefault
      .FormatFont 22, True, True
      .Insert "Welcome to Word OLE Automation"
      .InsertPara
      .FormatFont 10, False, False
      .Insert "Report Created:"
      .InsertDateTime "YYYY MM DD HH:MM:SS", False
      .InsertPara
    End With
  ?>
</OnKeyEnter>
```

A script block starts with “<?Script” followed with the script language identifier and ends with “?>”. Lines in between are script lines that must comply with the chosen script language. In the above example when the user pushes the Enter key the script will startup Microsoft Word, maximize the application and create a new document. Then it inserts some text in different font sizes.

Below is the same functionality but written in JavaScript.

```
<OnKeyEnter>
  <?Script JavaScript
    var ThisWord = new ActiveXObject("Word.Basic");
    with (ThisWord) {
      AppMaximize();
      FileNewDefault();
      FormatFont(22,1,1);
      Insert("Welcome to Word OLE Automation");
      InsertPara();
      FormatFont(10,0,0);
      Insert("Report Created:");
      InsertDateTime("YYYY MM DD HH:MM:SS",0);
      InsertPara();
    }
  ?>
</OnKeyEnter>
```

A good place to find instructions for programming Visual Basic script and JavaScript is [Microsoft MSDN](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vtoriMicrosoftWindowsScriptTechnologies.asp). At the time of this writing the direct link is

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vtoriMicrosoftWindowsScriptTechnologies.asp>

2. Script Calls

Instead of placing the script code inside an action you can create a subroutine and call it with a “CallScript” tag.

```
<OnKeyEnter>
  <CallScript Name="StartWord">
    <Argument>A CallScript Example</Argument>
    <Argument>12</Argument>
  </CallScript>
</OnKeyEnter>
```

The “Name” attribute specifies the subroutine to be called. Nested inside the “CallScript” tag are “Argument” tags - one for each of the subroutine parameters. A subroutine with no parameters looks like this

```
<OnKeyStop>
  <CallScript Name="CloseWord"/>
</OnKeyStop>
```

A subroutine is defined outside the “Events” tag in a global script block as illustrated in figure 1. F.ex the StartWord subroutine could look like this:

```
<?xml version="1.0" encoding="UTF-16"?>
<Module>
  <?Script VBScript
    Dim ThisWord

    Sub StartWord(Txt,Sz)
      Set ThisWord = CreateObject("Word.Basic")
      With ThisWord
        .AppMaximize
        .FileNewDefault
        .FormatFont Sz, True, True
        .Insert Txt
        .InsertPara
        .FormatFont 10, False, False
        .Insert "Report Created:"
        .InsertDateTime "YYYY MM DD HH:MM:SS", False
        .InsertPara
      End With
    End Sub
  ?>
  ...
```

Notice that the variable declaration of “ThisWord” is outside the subroutine. When a variable is placed outside a subroutine in a global script block it becomes a global variable that can be used in other subroutines and script actions.

3. Action Tags

Action tags are commands. They provide some of the common tasks needed in psf-files. They also provide another level of programming psf-files because they are easier to use than Visual Basic Script and JavaScript. It's possible to build a basic script module only using action tags. Some of the more advanced feature like using automation objects are only available when using scripts.

Currently the following action tags are defined

Tag	Tag value	Attributes	Function
Sleep		Time	Pause execution
SendKeys		Keys, {Ctrl, Alt, Shift}	Emulates keyboard input
SendMessage		Msg, WindowName ClassName, {wParam, lParam}	Sends a message to a window
SetMenuVisible	true false		Shows og hides the Palmbutler menu
RunProgram		Filename, {Arguments}	Executes an application
AppActivate		WindowName ProcessId	Assures the specified window has focus
ShowMessage	<i>Text</i>	{Time}	Shows a message
StoredVolume	true false		Restore/store volume on script entry/exit. Default false
VolumeUp		{Step}	Increments the volume by Step. Step defaults to 10. The volume can increase to a maximum of 100
VolumeDown		{Step}	Decrements the volume by Step. Step defaults to 10. The volume can decrease to a minimum of 0

Attributes inside {} are optional. The | indicates either/or.

ShowMessage shows a message on the screen.

```
<ShowMessage Time="2000">Hello World</ShowMessage>
```

This will show a message on the screen saying "Hello World!" for 2 seconds. The Time attribute defaults to 4 seconds if not specified.

The Sleep tag takes one argument – the duration in milliseconds.

```
<Sleep Time="1000"/>
```

Pauses execution for one second.

SendKeys sends a key sequence to the focused window. The optional attributes specifies whether Ctrl, Alt and Shift are pressed.

```
<SendKeys Ctrl="true" Keys="F"/>
```

This sends ctrl-F to the focused window.

```
<SendKeys Ctrl="true" Alt="true" Keys="{F1}"/>
```

This sends ctrl-alt-F1. The "Keys" attribute uses {} to specify virtual keys. This tag follows the convention used by the Windows Script Host SendKeys method. To learn more about how to specify special keys follow this link:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsmthsendkeys.asp>

or search for "SendKeys" at [Microsoft MSDN](#).

AppActivate is often used together with SendKeys.

```
<AppActivate WindowName="Calculator"/>  
<SendKeys Keys="{del}"/>
```

This assures that the right window has focus before sending keys.

SendMessage sends a window message

```
<SendMessage Msg="16" WindowName="Calculator"/>
```

This sends message 16 (WM_CLOSE) to the window with title "Calculator"

SetMenuVisible is used to hide and show the Palmbutler menu.

```
<SetMenuVisible>false</SetMenuVisible>
```

This hides the menu.

RunProgram runs an executable file

```
<RunProgram Filename="Notepad" Arguments="c:\autoexec.bat"/>
```

This starts notepad with the argument "c:\autoexec.bat"

Module tags

Tags placed under the module tag are settings. Currently the following tags are defined:

Tag	Tag value	Attributes	Function	
Description	<i>Text</i>		A description of the module	Required
RunWithShadow	true false		Allows a shadow module to run in the background	Optional, default is true
ShadowOverride	true false		Specifies that this modules keyevents overrides the keyevents of a shadow module	Optional, default is true
Extension	<i>Ext</i>	Type	This module supports files with extension <i>Ext</i>	Optional
Standalone	true false		This module can be started directly as a script item	Optional, default is true

The Description tag defines the description which is presented for the end user. Fx

```
<Description>Telephone book v1.0 by John Doe</Description>
```

could be a module for easy phone number lookup. If the pc has a modem connected the module could even dial a number from the Telephone book.

The RunWithShadow tag defines whether a module in shadow mode are allowed to run in the background. Currently only the audio module are able to run in shadow mode. The shadow mode is actually the default mode of the audio module . Only when audio animation is activated the audio module runs in normal mode. When the audio module is in shadow state it can play music while you are watching pictures, looking though the menu, running power point presentations or browsing the internet. The video and DVD module have defined RunWithShadow=false because they need the audio channel. In the above Telephone book example it would be obvious to set RunWithShadow=true in order to allow the user to listen to music while he works with the phone book. On the other hand if your were building

```
<Description>Quick Time Player v1.0 by John Doe</Description>
```

it would be obvious to set RunWithShadow=false.

The ShadowOverride tag tells which module has first priority for the key events.

```
<ShadowOverride>true</ShadowOverride>
```

This gives the script module first priority. Any keyevents not used by the script module are sent to the shadow module. If ShadowOverride=false the shadow module have first priority.

The Extension tag defines which file types the script module supports. Fx. for the previous Quick Time example the following Extension tags might be appropriate.

```
<Extension type="video">mov</Extension>  
<Extension type="video">mpg</Extension>  
<Extension type="video">mpeg</Extension>
```

When a script module defines Extension tags then it can be invoked through association. For this to work the script module must be registered as a plugin in the PalmButler Configuration. For example the Quick Time Player are started when the user activates any file with the extension "mov" provided that the user have chosen the Quick Time plugin as default handler for the file type "mov".

The Standalone tag indicates whether it makes sense to start the script directly. For example

```
<Standalone>>false</Standalone >
```

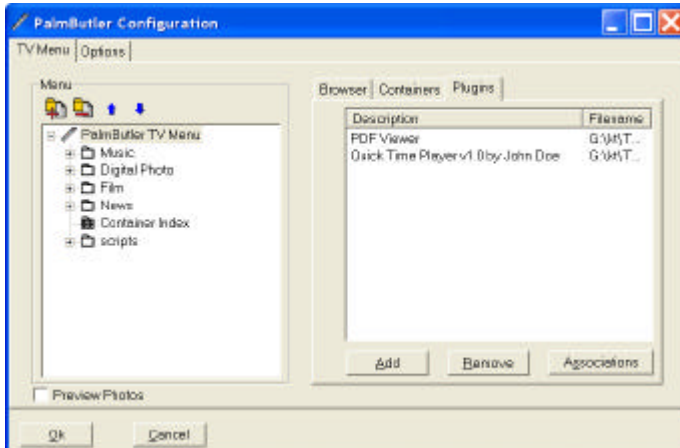
means that the script only should be started through file association. This is probably the appropriate setting for the Quick Time Player. But for the module

```
<Description>BabyCam v1.0 by John Doe</Description>
```

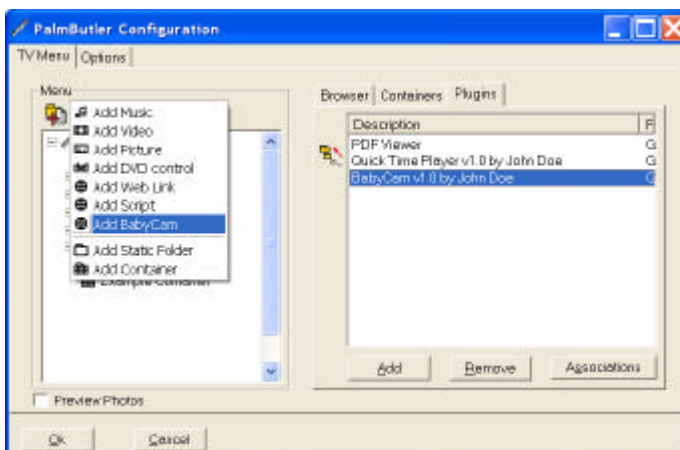
the correct setting is Standalone=true because this module only depends on input from a camera and microphone in the baby's chamber.

Plugins

Plugins are script modules which can register with the PalmButler system. In PalmButler Configuration on the tabsheet “Plugins” you can see a list of installed plugins. You can register new plugins and remove installed plugins.



It's not necessary to register a script module as a plugin in order to use it but plugins have some important advantages. They can be associated with one or more file types through the Extension tag and they can introduce new file types previously unknown to the PalmButler system. After adding a module with a new file type the user is able to browse, drag and drop the new file type to his menu and use it throughout the system as any predefined file type. Standalone plugins can be added to the PalmButler TV menu through the “Add menu item” button as shown below.



For example, the standalone “BabyCam” plugin are shown above. The two other plugins are not in the Add menu because they have the Standalone tag set to false.

The Script Objects

Palmbutler publishes a number of objects for script access. They make it possible for the script module to iterate through folders and items in the menu, changing settings and issuing commands to the Palmbutler system.

Object	Function
pbMenu	The root object. Represents the main menu. Also gives access to global settings and functions
pbFolder	Represents a folder node in the menu
pbItem	Represents an item node in a folder
pbCollection	A list of folders, items or popup menu items. The list can contain a mix of these object types
pbPopupMenu	Represents a popup menu.
pbPopupMenuItem	Represents a popup menu item

The pbMenu object is global and created beforehand, so it is ready for script access immediately. Currently pbMenu has the following methods and properties.

pbMenu properties	Type	Description	Access
CurFolder	pbFolder	The folder currently open in the menu	Read only
CurItem	pbItem	The item currently selected in the menu	Read only
Folders	pbCollection	A collection of folders (pbFolder) in the main menu	Read only
Visible	boolean	Read or set whether the menu is visible or not	Read/Write
MasterVolume	integer	Access to the master volume in the system volume control. Valid values 0 - 100	Read/Write
WaveVolume	integer	Access to the wave volume in the system volume control. Valid values 0 - 100	Read/Write
StoredVolume	boolean	Restore/store volume on script entry/exit. Default false. If set to true Palmbutler will remember the volume for next time this script is activated	Read/Write

pbMenu methods	Type	Description
Extract	pbCollection	Queries the menu for specific items and folders
ItemById	pbItem	Searches all items and folders for the specific item
CreatePopupMenu	pbPopupMenu	Creates a popup menu object

The pbFolder object has the following members

pbFolder properties	Type	Description	Access
Folders	pbCollection	The subfolders. A collection of pbFolder	Read only
Items	pbCollection	The items in the folder. A collection of pbItem	Read only
Name	string	Folder name	Read/Write
URL	string	The folder path/url. Used in dynamic folders and containers	Read/Write
Typ	integer	Folder type. See list of item types	Read only
ItemId	integer	Uniq ID among all folders and items in the menu	Read only

pbFolder methods	Type	Description
Extract	pbCollection	Queries the folder for specific items and folders

The pbItem object has the following members

pbItem properties	Type	Description	Access
Name	string	Item name	Read/Write
URL	string	The item path/url	Read/Write
Skip	boolean	If true the user wants to ignore this item in play lists	Read/Write
Typ	integer	Item type. See list of item types	Read only
ItemId	integer	Uniq ID among all folders and items in the menu	Read only

The pbCollection object has the following members

pbCollection properties	Type	Description	Access
Count	integer	Number of items in the collection	Read only

pbCollection methods	Type	Description
Item	pbItem pbFolder pbPopupMenuItem	Indexed access to each item. First item has index 0

Here are some examples:

```
<?Script VBScript
  Dim N

  N = pbMenu.CurItem.Name
  MsgBox N,0,"Hello"
?>
```

This script stores the name of the current item i variable N. It then pops up a message box showing the item name with caption "Hello"

```
<?Script VBScript
  Dim N

  For Each Item in pbMenu.CurFolder.Items
    N = N & Item.Name & ", "
  Next

  MsgBox N
?>
```

This script iterates through the items in the current folder in the Palmbutler menu and adds all names to the variable N. Then it shows the list of names in a messagebox.

There exists a number of different types of items and folders. To distinguish between different types of items use the Typ property of the object

Typ	Description
Folders:	
itFolder	Static folder
itDynFolder	Dynamic folder
itConFolder	Container folder
itDynChildFolder	Dynamic subfolder
itConChildFolder	Container subfolder
Items:	
itDVD	DVD
itAudioClip	Audio Clip

itAudioStream	Audio Stream
itVideoClip	Video Clip
itVideoStream	Video Stream
itPhoto	Photo
itBrowserURL	BrowserURL
itPresentation	Presentation
itScript	Script
itOther	Other item

Each item and folder has a uniq id. Use the ItemId property as a reference to the item or folder. If you know an items Id you can find it using the ItemById method. Fx

```
<?Script VBScript
Dim PhotoFolder

Set PhotoFolder = pbMenu.ItemById(100003)

MsgBox PhotoFolder.Name
?>
```

This will find the item with id=100003 and display its name in a messagebox. Some of the preinstalled items have a fixed id.

Preinstalled	Symbol	Numeric value
Music folder	idMusicFolder	100001
Video folder	idVideoFolder	100002
Photo folder	idPhotoFolder	100003
News folder	idNewsFolder	100004
Container Index	idContainerIndex	100005
Jokebox folder	idJukeboxFolder	100006

So the example above will find the photo folder unless it has been deleted.

The URL property can either contain an internet address or a locale file path. The Extract method extracts a list of specific item types. The list can optionally be sorted.

Extract method parameters	type	Comments
ItemTypes	array	Required
Recursive	0 1	Optional, default = 1
Disabled	0 1	Optional, default = 0
Sorting	integer	Optional, default = 4
SortTag	string	Optional, default = ""
Ascending	0 1	Optional, default = 1
Sorting2	integer	Optional, default = 4
SortTag2	string	Optional, default = ""
Ascending2	0 1	Optional, default = 1

Most of the parameters are optional, so it can be called with a single argument or with as many as 9 arguments. The ItemTypes parameter is an array of Item types to search for.

```
<?Script VBScript
Dim A(1)
Dim ItemList

A(0) = itPhoto
```

```

A(1) = itPresentation
Set ItemList = pbMenu.Extract(A)
?>

```

This script declares an array with 2 elements A. It then stores the photo type (itPhoto) in the first array element and presentation type (itPresentation) in the second element. The last line extracts all photo and presentation items in the Palmbutler menu to the variable ItemList. The ItemList can then be used in a For Each statement to iterate through the items.

The Extract method is also available in folder objects. Called from a folder object only items in this folder are searched. The parameter Recursive decides if subfolders are searched as well.

The Disabled parameter decides if Items marked with Skip=true are included in the search. Users can disable an item from the Palmbutler menu by setting the Skip option.

The Extract method can sort the resulting list from 1 or 2 sort criteria.

Sort & Sort2	Description
fsNone	No sorting
fsByName	By name
fsByType	By media type
fsByTag	By tag. Tag specified in SortTag and SortTag2
fsDefault	Sorting chosen by the user

Use one of these values in the Sort and Sort2 parameter. The SortTag and SortTag2 are used if you choose to sort by tag. Tags are extra information read from mp3 files (id3tags) and containers. Music files often contains the following tags:

Artist
Album
Genre
Composer
Year
Language

The following code finds all music items in the music folder and sort them after Artist and secondary after Name in ascending order.

```

<?Script VBScript
Dim A(1)
Dim MusicFolder
Dim ItemList

A(0) = itAudioClip
A(1) = itAudioStream

Set MusicFolder = pbMenu.ItemById(idMusicFolder)
Set ItemList = MusicFolder.Extract(A,1,0,sByTag,"Artist",1,sByName,"",1)
?>

```

Popup Menus

The pbFolder object has the following members

pbPopupMenu properties	Type	Description	Access
Items	pbCollection	The items in the popup menu. A collection of pbPopupMenuitem	Read only

pbPopupMenu methods	Type	Description
AddItem	pbPopupMenuitem	Adds an item to the menu
Show		Show the menu on the screen. The menu takes control of keyboard events

pbPopupMenu events	Event arguments	Description
OnChange	pbPopupMenuitem	This event is triggered when the user changes something in the menu
OnEnter	pbPopupMenuitem	This event is triggered when the user activates a menu item with the enter key
OnClose		The event is triggered when the popup menu closes

The pbPopupMenuitem object has the following members

pbPopupMenuitem properties	Type	Description	Access
Caption	string	The menu item text	Read/Write
Typ	integer	Menu item type	Read/Write
Id	integer	Item id. Used to identify the item fx in events	Read/Write
Value	integer	Item value. It's meaning depends on Typ	Read/Write
MinValue	integer	The minimum allowed value for Value. Used by menu item type: mtInteger	Read/Write
MaxValue	integer	The maximum allowed value for Value. Used by menu item type: mtInteger	Read/Write
SingleStep	integer	The amount Value is incremented/decremented pressing KeyRight or KeyLeft. Used by menu item type: mtInteger	Read/Write
RepeatStep	integer	The amount Value is incremented/decremented when holding down KeyRight or KeyLeft. Used by menu item type: mtInteger	Read/Write
SubMenu	pbPopupMenu	A sub menu. Used by menu item type: mtSubMenu	Read only

pbPopupMenuitem methods	Type	Description
AddChoice		Adds a choice to the list of possible choices for the menu item. Used by menu item type: mtSelection

There exists 6 types of popup menu items

Typ	Description
mtAction	Asks the user to press the enter key
mtSelection	Gives the user a list of choices to choose from
mtInteger	The user can change an integer value using KeyRight and KeyLeft
mtCheck	The user can check or uncheck these items
mtArrow	Asks the user to press either KeyLeft or KeyRight
mtSubMenu	Opens up a sub menu

A popup menu is typically used together with the OnKeyMenu event. When the user presses the menu key a popup menu gives the user access to various options and actions. The following gives an example defining a popup menu and responding to menu events. The example assumes the following constants have been defined in a global script block

```

<?Script VBScript
  Const miZoom = 1
  Const miPage = 2
  Const miOpen = 3
  Const miNextDoc = 4
  Const miPrevDoc = 5
?>

<OnKeyMenu>
<?Script VBScript
  Dim MI, MI2

  Set PopupMenu = pbMenu.CreatePopupMenu
  Set PopupMenu.OnChange = GetRef("PopupMenuChange")
  Set PopupMenu.OnEnter = GetRef("PopupMenuEnter")
  Set PopupMenu.OnClose = GetRef("PopupMenuClose")

  Set MI = PopupMenu.AddItem
  MI.Caption = "Zoom"
  MI.Type = mtSelection
  MI.Id = miZoom
  MI.Value = 1
  MI.AddChoice("50%")
  MI.AddChoice("100%")
  MI.AddChoice("200%")
  MI.AddChoice("300%")
  MI.AddChoice("400%")
  MI.AddChoice("500%")

  Set MI = PopupMenu.AddItem
  MI.Caption = "Page"
  MI.Type = mtInteger
  MI.Id = miPage
  MI.MinValue = 1
  MI.MaxValue = 10
  MI.Value = 1

  Set MI = PopupMenu.AddItem
  MI.Caption = "Open"
  MI.Type = mtSubMenu
  MI.Id = miOpen

  Set MI2 = MI.SubMenu.AddItem
  MI2.Caption = "Next Document"
  MI2.Type = mtAction
  MI2.Id = miNextDoc

  Set MI2 = MI.SubMenu.AddItem
  MI2.Caption = "Previous Document"
  MI2.Type = mtAction
  MI2.Id = miPrevDoc

  PopupMenu.Show
?>
</OnKeyMenu>

```

For each AddItem a new line is added to the menu. After defining the menu the method PopupMenu.Show activates the menu on the screen. It gives the following result



when the user right clicks on the “Open” item. The menu automatically calculates its size to contain the texts and menu item types.

To catch the user selections you need to define one or more events and assign them to the event properties of the popup menu. The event routines should be placed in a global script block. The OnChange event routine and the OnEnter event routine are called with one argument – the active menu item. You can use this argument to decide which item is active and the state of the item. The Id property identifies the item and the Value property tells you what the user have selected. The OnClose event routine have no arguments.

```
<Module>
  <?Script VBScript
    Sub PopupMenuChange(MI)
      If MI.Id = miZoom Then
        ...Do something
      ElseIf MI.Id = miPage Then
        ...Do something
      End If
    End Sub

    Sub PopupMenuEnter(MI)
      If MI.Id = miNextDoc then
        ...Do something
      ElseIf MI.Id = miPrevDoc then
        ...Do something
      End If
    End Sub

    Sub PopupMenuClose
      ...Do something
    End Sub
  ?>
```

See the PDF Viewer plugin for an example of using script objects.

Global Script Functions

Palmbutler publishes a number of global functions directly available in script blocks.

Function name	Description	Arguments	Return value
FindWindow	Retrieves the handle to the top-level window whose class name or window name match the specified strings.	Classname, WindowName	Handle integer
SendMessage	Sends the specified message to a window	hWnd, Msg, {wParam, lParam}	IResult integer
AppActivate	Activates an application window.	WindowName ProcessId	Success boolean
SendKeys	Sends one or more keystrokes to the active window (as if typed on the keyboard).	Keys	-
Exec	Executes an application		WshScriptExec object
Sleep	Pause execution	Time	-
keybd_event	Synthesizes a keystroke	bVk, bScan, dwFlags, dwExtraInfo Consult http://msdn.microsoft.com for details	-
mouse_event	Synthesizes mouse motion and button clicks.	dwFlags, dx, dy, dwData, dwExtraInfo Consult http://msdn.microsoft.com for details	-

These functions calls Windows OS functions and detailed description of each function are available at [Microsoft MSDN](http://msdn.microsoft.com).

Here are an example starting the Winamp player.

```
<OnInitialization>
  <?Script VBScript
    Dim hWnd, LResult

    pbMenu.Visible = false
    Set Winamp = Exec("%programfiles%\Winamp\winamp.exe")
    Sleep 200
    hWnd = FindWindow(WinampClass,"")
    LResult = SendMessage(hWnd,WinampMsg,WinampStart)
  ?>
</OnInitialization>
```

This Script first hides the Palmbutler menu. Then it starts winamp assuming the default path in the program files directory. It then waits 200 ms. The Winamp main window handle is returned in hWnd. SendMessage sends the startplaying command to Winamp using the Window handle.

For this to work the following constants must be defined in a global script block.

```
<?Script VBScript
  Const WinampClass = "Winamp v1.x"
  Const WinampMsg = 273
  Const WinampStart = 40045
  Const WinampPlay = 40046
  Const WinampStop = 40047

  Dim Winamp
?>
```