Table of Contents

Visual programming of dialog boxes is quick and accurate. Use generic variable names so you can reuse your favorite dialogs.

The WIL Dialog Editor (see Filenames: Appendix A, page **Error! Bookmark not defined.** for filename) provides a convenient method of creating dialog box templates for use with the **Dialog** function.

It displays a graphical representation of a dialog box, and allows you to create, modify, and move individual controls which appear in the dialog box.

After you have defined your dialog box, the Dialog Editor will generate the appropriate WIL code, which you can save to a file or copy to the Clipboard for pasting into your WIL program.

Note: The WIL Dialog Editor comes with an on-line help file (For the name of the help file see Filenames: Appendix A, page **Error! Bookmark not defined.**), as well as detailed instructions in the next section. Simply select the Help function in the Dialog Editor for detailed instructions on using the program.

You can have as many as 100 controls in a WinBatch dialog. However, too many controls can be confusing. Aim for simple dialogs with a consistent appearance between different ones.

The WIL Dialog Editor offers quick production of custom dialog boxes for your WinBatch programs.

The WIL Dialog Editor allows you to create dialog box templates for WIL using the WDL format. The Dialog Editor will write the WIL script statements necessary to create and display the dialog.

You can visually design your dialog box on the screen and then save the template either to a .WDL file or the Windows Clipboard.

You can include the dialog template code directly in your batch code, or you can use the batch language "Call" command to execute the dialog template. For example:

```
Call("Sample.WDL", "")
```

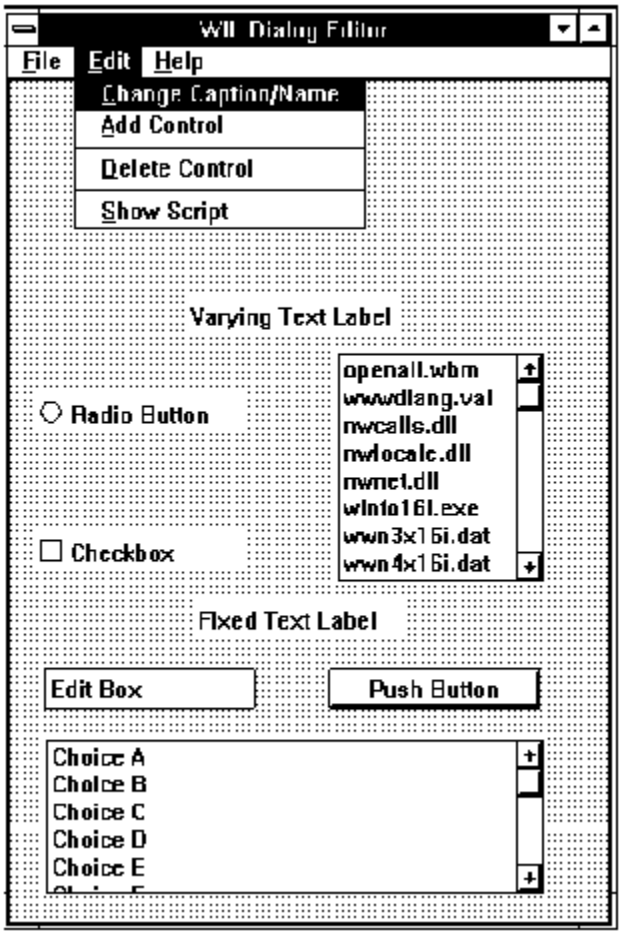
Using the Dialog Editor is easy. Once it is loaded, these hints offer a quick way to become comfortable with dialog box construction.

The dialog editor filename for 16 bit Windows use is: wwd1g16i.exe.

Launch the dialog editor executable, (see Filenames: Appendix A on page **Error! Bookmark not defined.** for filename).

The editor will look like the following:

To control the size of your dialog box, resize the WIL Dialog Editor. Your dialog will be the same size as this editor's window.



There are three standard menus in this program; FILE, EDIT, and HELP.

File
Edit
Help

New
When you select New, any currently loaded template will be discarded and the slate will be clean for a new dialog. You will be prompted to enter the caption (title) for your dialog box, and a WIL variable name used to refer to the dialog box in the WIL scripts.

Load
Loads a dialog template from a file.

Save
Saves a dialog template to the current file.

Save As
Saves the dialog template to a file using a different filename.

Load from Clipboard
Loads a dialog template from the Windows Clipboard.

Save to Clipboard
Saves the dialog template to the Windows Clipboard.

Change Caption/Name
Allows you to change the Dialog caption (title) and/or the variable name used to refer to the dialog.

Note: Left Mouse double-clicking the dialog box background will also execute this menu item.

Add Control
Adds a new control to your dialog template.
Note: Right Mouse double-clicking has the same effect.

Delete Control
Surprisingly enough, Delete Control does not actually delete a control. It just reminds you how to do it. To delete a control, position the mouse cursor over the control and press the delete key.

Show Script
Displays the WIL script generated during the dialog edit session. Once you learn how the dialog scripts operate, viewing the script is a quick way to scan for errors. You will notice that some script lines cannot be viewed in their entirety, in which case simply double click it to view the entire line.

Index
Displays the Index of the On-line help information.

Menu Commands
Displays information about the WIL Dialog Editor menu commands.

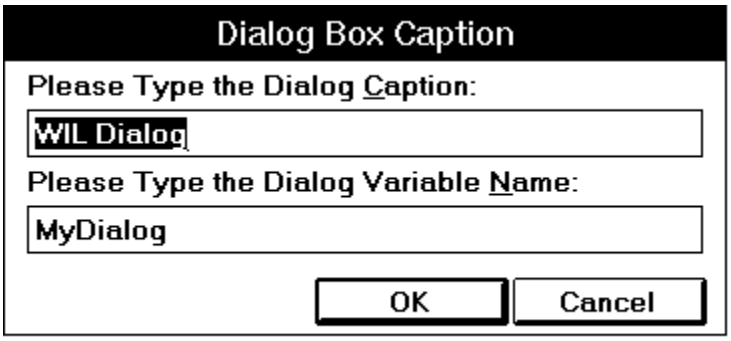
How to use Help
Activates the Microsoft Windows Index to Using Help.

About
Displays the WIL Dialog Editor About dialog which includes the version number of the program.

Double click with the left mouse button on the workspace background to display the caption box.

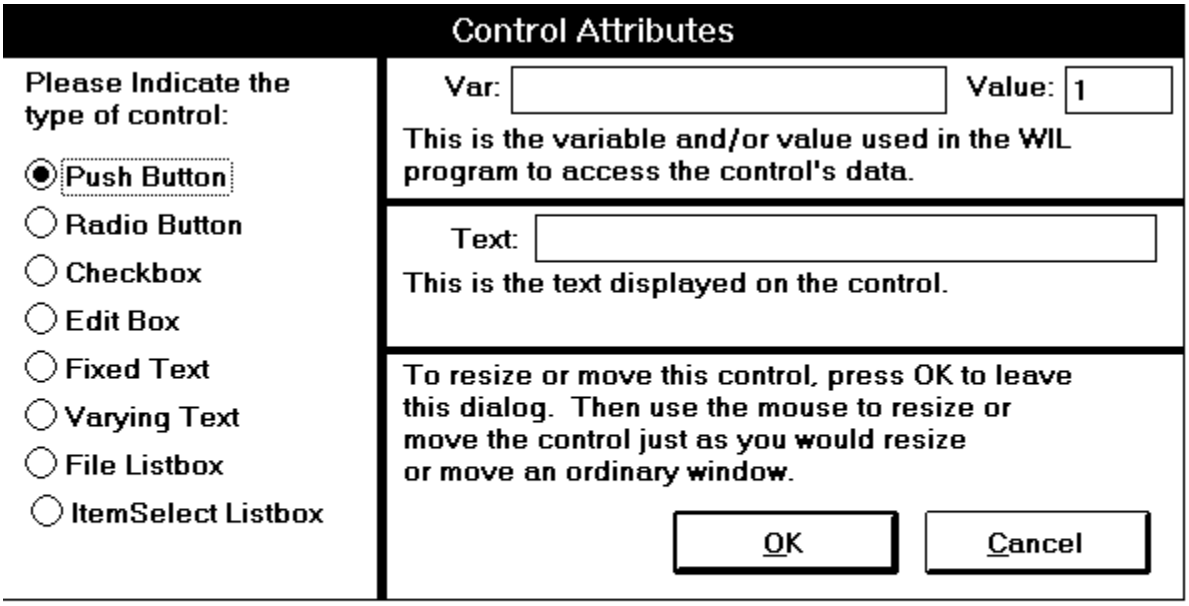
The **Dialog Caption** is the title of the dialog box as it appears in the title bar. The **variable name** is the name of the dialog as seen in the script.

This information can be entered or changed at any time. However, we suggest filling it whenever you start a new dialog box. To change the caption double click on the workspace, (not on a control) with the left mouse button.



To add a control, double click with the right mouse button where you want the control. Fill in the information in resulting dialog box about the control.

Choose the control on the left and fill in the appropriate attributes on the right. The control may need a **Variable** name, a **Value** or **Text**. Not all information will be needed for each control. Fill in only the items which are not grayed out.



The following table is a quick reference of what attributes are required for each control.

Control	Variable	Value	Text
Push Button		✓	✓
Radio Button	✓	✓	✓
Check Box	✓	✓	✓
Edit Box	✓		✓
Fixed Text			✓
Varying Text	✓		✓
File Listbox	✓		
ItemSelect Listbox	✓		

To **MOVE** the control, click on it and drag it to a new position with the left mouse button.

To **SIZE** a control, click on the edge and drag with the left mouse button.

To **DELETE** a control, position the mouse over the control and press the delete key.

Once you are happy with your work, choose **"Save"** or **"SaveAs"** from the **File** menu to save your work to a file. Choose **"Save to Clipboard"** to put the work into the clipboard so that it can be easily pasted into one of your WIL scripts.

Take a peek at the resulting script with the **File "ShowScript"** command to begin to get used to what WIL Dialog Scripts look like.

See: Decipher the Script

Here is an example of what a WIL Dialog Editor script looks like. For information on what it all means, see Decipher the Script.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

The Dialog Editor follows a specific format when creating your script. For example, here is a dialog box script we created.

The first line sets the format and specifies the version of the Dialog Editor being used.

```
ExampleFormat=`WWWDLGED,5.0`
```

The next section establishes the caption which will appear in the title bar of the dialog box along with the coordinates, size and number of controls in the dialog box.

```
ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12
```

The third section contains the code for the actual controls. Each line has specific information.

```
Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

When the first line in the example above is broken down, the parts are as follows.

Code	**Definition**
Example	Dialog Variable Name
01	Control Number
27,113,76,DEFAULT	Coordinates of the control
PUSHBUTTON	Control Type
"DEFAULT",	Variable name
OK	Text
1	Value

Each Dialog script will end with the following line, making it easy to test the PushButton return values.

```
ButtonPushed=Dialog("Example")
```

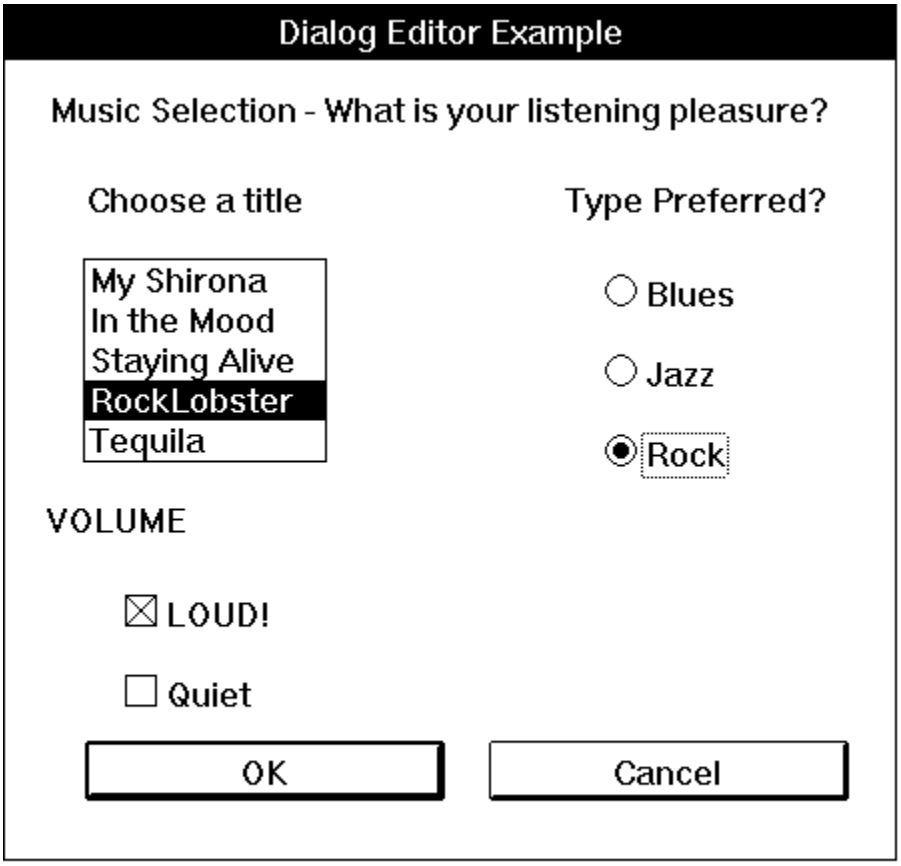
Put all the parts together and the completed script looks like the following.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
```

```
                                                    your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Note:

Here is the completed dialog box.



Some of the Controls require extra knowledge or special handling.

- Push Button
- Radio Button
- Check Box
- Edit Box
- Fixed Text
- Varying Text
- File Listbox
- ItemSelect Listbox

Any information which is needed by the Dialog Box Controls should be set up in the script prior to the dialog code. By setting the variables, you can pass lists, files, and set which options are chosen by default.

When creating **Push Buttons**, it is a good idea to assign the value of 1 to your "OK" button equivalent and 0 to your "Cancel" button equivalent. Each button will have a separate value. The Dialog Editor adds a line to the end of your script which helps to test return values.

```
Buttonpushed=Dialog"MyDialog"
```

To test a return value do the following:

```
If Buttonpushed == 1 then goto label
```

"Cancel" or the value 0 will generally look for a label **:cancel**. If not found, it will exit.

For more information, see **Things to Know** in the **WIL Reference Manual**.

Used in situations to choose one item over another. You can have 9 choices per variable.
In using a **Radio Button**, the variable assigned is the same for each of the choices but the value is different.
For example, the script in a Dialog may look like:

```
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

The variable "music" is the same on both lines. The text and the values are different on each line.

Note: **Radio Button** cannot have a value of 0.

Offers a choice of options. Any number may be marked or left unmarked. Each Check Box has its own specific information. Variable, Value and Text are different, allowing the user to choose more than one.

Use this control to create a box in which a choice can be entered by default and then altered by the user.

Use Fixed Text to display labels, descriptions, explanations, or instructions. The Control Attribute box will let you type an endless amount of information into the text box. However, only about 60 characters will be displayed.

Use Varying Text to grab data which may change, like a date or a password, from somewhere else.

Use File Listbox to allow the user to choose a file from a list box. Set your variable to display a directory path and filemask or the result of **FileItemize**.

```
wbtfiles="C:\WINBATCH\*.WBT"
wbtfiles=FileItemize("*.bak")
```

This box can be tied with the variable to an Edit Box or to Fixed Text. When the user chooses a file, it will be displayed in the Edit Box or in the place of Fixed Text if the variable is the same.

Note: When File Listbox is used, the dialog editor assumes that a file must be chosen before it proceeds. Add the following WIL command to the top of your script if you wish to allow the dialog to proceed without a file selection.

```
IntControl(4, 0,0,0,0)
```

See the WIL manual for more information on **IntControl**.

Use the ItemSelect Listbox to allow the user to choose an item from a list box. This option is similar to the WIL commands **AskItemList**, and **ItemSelect**. Set your variable to display a list of items delimited by a tab.

Use **@tab,** a predefined constant, as the delimiter.

```
tunes="My Shirona%@tab%In the Mood%@tab%Staying Alive%@tab%
                              RockLobster%@tab%Tequila"
```

Note: When an ItemSelect Listbox is used, the dialog editor assumes that a file must be chosen before it proceeds. Add the following WIL command to the top of your script if you wish to allow the dialog to proceed without a file selection.

```
IntControl(4, 0,0,0,0)
```

See the WIL manual for more information on **IntControl**.

yesTRUEyesyesyesHELLLP! Generatedextjumpsyesyes13/10/94

Table of Contents

Help file produced by **HELLLP!** v2.3a , a product of Guy Software, on 10/13/1994 for WILSON WINDOWWARE, INC..
The above table of contents will be automatically completed and will also provide an excellent cross-reference for context strings and topic titles. You may leave it as your main table of contents for your help file, or you may create your own and cause it to be displayed instead by using the I button on the toolbar. This page will not be displayed as a topic. It is given a context string of _:_ and a HelpContextID property of 32517, but these are not presented for jump selection.
HINT: If you do not wish some of your topics to appear in the table of contents as displayed to your users (you may want them ONLY as PopUps), move the lines with their titles and contexts to below this point. If you do this remember to move the whole line, not part. As an alternative, you may wish to set up your own table of contents, see Help under The Structure of a Help File.
Do not delete any codes in the area above the Table of Contents title, they are used internally by HELLLP!

Note:

The songs that appear in the ItemSelect Listbox are listed earlier in the script on one continuous line as the variable, tunes.
ie.

```
tunes="My Shirona%@tab%In the Mood%@tab%Staying Alive%@tab%
                                    RockLobster%@tab%Tequila"
```

Variables can be defined above the dialog script or in another WBT file above the statement which calls the dialog file.

View the Script

Take a peek at the resulting script with the **File "ShowScript"** command to begin to get used to what WIL Dialog Scripts look like.

See: Decipher the Script

ShowScript

Here is an example of what a WIL Dialog Editor script looks like. For information on what it all means, see Decipher the Script.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Decipher the Script

The Dialog Editor follows a specific format when creating your script. For example, here is a dialog box script we created.

The first line sets the format and specifies the version of the Dialog Editor being used.

```
ExampleFormat=`WWWDLGED,5.0`
```

The next section establishes the caption which will appear in the title bar of the dialog box along with the coordinates, size and number of controls in the dialog box.

```
ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12
```

The third section contains the code for the actual controls. Each line has specific information.

```
Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
```

When the first line in the example above is broken down, the parts are as follows.

Code	Definition
Example	Dialog Variable Name
01	Control Number
27,113,76,DEFAULT	Coordinates of the control
PUSHBUTTON	Control Type
"DEFAULT",	Variable name
OK	Text
1	Value

Each Dialog script will end with the following line, making it easy to test the PushButton return values.

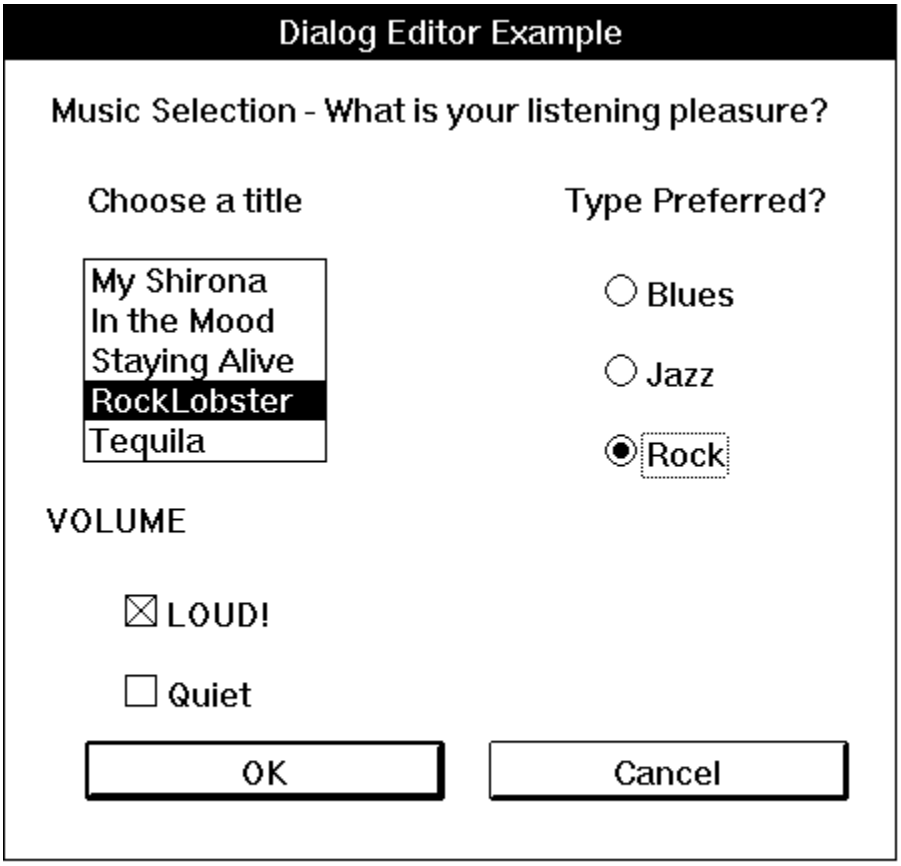
```
ButtonPushed=Dialog("Example")
```

Put all the parts together and the completed script looks like the following.

```
ExampleFormat=`WWWDLGED,5.0`

ExampleCaption=`Dialog Editor Example`
ExampleX=120
ExampleY=50
ExampleWidth=179
ExampleHeight=160
ExampleNumControls=12

Example01=`16,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"OK",1`
Example02=`97,136,72,DEFAULT,PUSHBUTTON,DEFAULT,"Cancel",0`
Example03=`120,40,48,DEFAULT,RADIOBUTTON,music,"Blues",1`
Example04=`120,56,56,DEFAULT,RADIOBUTTON,music,"Jazz",2`
Example05=`120,72,56,DEFAULT,RADIOBUTTON,music,"Rock",3`
Example06=`24,104,112,DEFAULT,CHECKBOX,volume,"LOUD!",1`
Example07=`24,120,104,DEFAULT,CHECKBOX,volume2,"Quiet",2`
Example08=`8,88,64,DEFAULT,STATICTEXT,DEFAULT,"VOLUME"`
Example09=`9,6,164,DEFAULT,STATICTEXT,DEFAULT,"Music Selection - What is
                                              your listening pleasure?"`
Example10=`16,40,48,40,ITEMBOX,tunes,DEFAULT`
Example11=`112,24,56,DEFAULT,STATICTEXT,DEFAULT,"Type Preferred?"`
Example12=`16,24,49,DEFAULT,VARYTEXT,song,"Choose a title"`

ButtonPushed=Dialog("Example")
```

Note:

Here is the completed dialog box.



WILSON WINDOWWARE ORDER FORM

Name: _____

Company:_____

Address:_____

_____

City: _____ **St:**_____ **Zip:**_____

Phone: (_____)_____ **Country:**_____

____ WinBatch @ $69.95 : _____.____

____ WinBatch Compiler @$395.00 : _____.____

____ WinBatch 32 @ $99.95 : _____.____

____ WinBatch 32 Compiler @$495.00 : _____.____

____ LetterBox Pro for Win 3.1 @$195.00 : _____.____

Upgrades

____ WinBatch to WinBatch 32 @ $30.00 : _____.____

____ WinBatch Compiler to WinBatch 32 Compiler @$100.00 : _____.____

Shipping

____ US and Canada shipping @ $5.00 : _____.____

____ Foreign air shipping (except Canada) @ $12.50 : _____.____

Total: _____.____

Please enclose a check payable to Wilson WindowWare or you may use Access, Amex, Visa, MasterCharge, or EuroCard. For credit cards, please enter the information below:

Card #:__ __ __ __ - __ __ __ __ - __ __ __ __ - __ __ __ __ **Expiration date: ____/____**

Signature: _____

Where did you hear about or get a copy of our products?

_____

International customers please see note on previous page.

Version 4.0 Example - Devadd.wbt

This is an example of adding "Device=" line(s) to the [386ENH] section of ;the SYSTEM.INI. It is still possible to use this example, however ;there is an alternative, preferred solution for Version 5.0.

EXAMPLE

```
windir = DirWindows(0)
sysini = "%windir%system.ini"
systmp = "%windir%system.tmp"

hIn = FileOpen(sysini, "READ")
hOut = FileOpen(systmp, "WRITE")
found = 0

:nextline
line = FileRead(hIn)
If line == "*EOF*" Then Goto done
FileWrite(hOut, line)
If found == 1 Then Goto nextline
If StriCmp(line, "[386ENH]") != 0 Then Goto nextline
found = 1

; here's where we add the new line(s)
FileWrite(hOut, "Device=DUMMY1.386")
FileWrite(hOut, "Device=DUMMY2.386")

Goto nextline

:done
FileClose(hIn)
FileClose(hOut)
If found == 1 Then FileCopy(systmp, sysini, @FALSE)
Then Message("DEVADD", "Operation complete")
Else Message("DEVADD", "[386ENH] section not found")
FileDelete(systmp)
```

Version 5.0 EXAMPLE - BinaryPokeStr

This example writes a new device= line to SYSTEM.INI. It is *very* fast

Example:

```
NewDevice = "DEVICE=COOLAPP.386"
;
; Change to the Windows Directory
DirChange(DirWindows(0))
;
; Obtain filesize and allocate binary buffers
fs1=FileSize("SYSTEM.INI")
srcbuf  = BinaryAlloc(fs1)
editbuf = BinaryAlloc(fs1+100)
;
; Read existing system.ini into memory
BinaryRead( srcbuf, "SYSTEM.INI")
;
; See if this change was already installed.  If so, quit
a = BinaryIndex( srcbuf, 0, "COOLAPP.386", @FWDSCAN)
if a != 0 then goto AlreadyDone
;
; Find 386Enh section.
a = BinaryIndex( srcbuf, 0, "[386Enh]", @FWDSCAN)
;
;
; Find beginning of next line ( add 2 to skip over our crlf )
cuthere = BinaryIndex( srcbuf, a, @CRLF, @FWDSCAN) + 2
;
; Copy data from beginning of file to just after [386Enh}
; to the edit buffer
BinaryCopy( editbuf, 0, srcbuf, 0, cuthere)
;
; Add the device= line to the end of the edit buffer, and add a CRLF
BinaryPokeStr( editbuf, BinaryEodGet(editbuf), strcat(NewDevice,@CRLF))
;
; Copy remaining part of source buffer to the edit buffer
a = BinaryEodGet(editbuf)
b = BinaryEodGet(srcbuf)
BinaryCopy( editbuf, a, srcbuf, cuthere, b-cuthere)
;
; Save file out to disk.  Use system.tst until it is
; completely debugged
BinaryWrite( editbuf, "SYSTEM.TST")
;
; Close binary buffers
:AlreadyDone
BinaryFree(editbuf)
BinaryFree(srcbuf)
```