# Table of Contents

# Chapter 1: What is "Dialog Toolkit"

"Dialog Toolkit" is a free AppleScript library created by Shane Stanly that allows scripters to create a dialog with more then 1 string of output text, more then 1 input field, more then 3 buttons as well as include popup buttons, checkboxes, radio buttons, any kind of label at all, as well as create something called a path control. An example of 1 type of a "path control" (NSPathStyleNavigationBar) is seen at the bottom of a Finder window to show the path of the currently selected item. "Dialog Toolkit" uses the NSPathStyleStandard type for the path controls.

In addition to providing new features "Dialog Toolkit" also makes the actual scripting easier to perform. A lot of the work necessary to create individual "items" to go in the alert, as well as creating the actual enhanced alert, are done by the "Dialog ToolKit."

The actual item used to display the information in an enhanced alert is something called an NSAlert. This is the basic building block programmers use to create Alerts. Using NSAlerts affects things like:

What the enhanced alert looks like,
What can go into the enhanced alert,
Operations that can be performed by an enhanced alert,
The way an enhanced alert is used,
What features that come built into an enhanced alert without additional coding, and
The key commands that are built into an enhanced alert.

All of the features just listed comes from the object oriented nature of Obj-C. Different objects in Obj-C have different capabilities and know about different things. So much of the way "Dialog Toolkit" behaves and what is possible to do with it comes from the behavior and power of the NSAlert object.

Therefore the better a person understands NSAlert the better they can understand "Dialog Toolkit." But "Dialog Toolkit" can still be used without knowing anything about NSAlerts. Therefore "Chapter 2: The basics about NSAlert" can be skipped if you only want to use "Dialog Toolkit" as is, and only want to use as is the 5 capabilities that are nonstandard in ordinary AppleScript dialogs (popup buttons, checkboxes, radio buttons, labels and path controls).

To tie it all together it can be said that using "Dialog Toolkit" allows the scripter to produce an enhanced alert while using ordinary NSAlerts, and read the user information entered after the dialog is dismissed.

This manual is not intended as a guide to NSAlerts used separate from "Dialog Toolkit." Using NSAlerts without "Dialog Toolkit" will most likely bring up issues not discussed in this manual.

# Chapter 2: The basics about NSAlert

Note: "This chapter can be skipped if you only want to use "Dialog Toolkit" as is, and only want to use buttons, checkboxes, radio buttons, labels and path controls as is. Changing any of these things in "Dialog ToolKit" without understanding the basics about the NSAlert would most likely lead to problems.

The programmer can specify 6 types of things:

- Alert style
- Alert text
- Button titles
- Custom icon
- Help text: Can can allow a help button to appear in the alert for help relating to the alert
- Suppression checkbox: Option to show/hide checkbox to suppress future alters

The default alert style is NSWarningAlertStyle.

## Alert styles

"Alert style" is a property of NSAlert

NSWarningAlertStyle (integer constant = 0)
Used to warn the user about a current or impending event. The purpose is more than informational but not critical.

NSInformationalAlertStyle (integer constant = 1)
An alert used to inform the user about a current or impending event.

NSCriticalAlertStyle (integer constant = 2)
Reserved for critical alerts, such as when severe consequences might result from certain user responses (for example, a “clean install” will erase all data on a volume). This style includes a caution icon badged with the app icon.

Currently there is no visual difference between informational and warning alerts. You should only use the critical alert style if warranted.

As of 8/10/2016 the NSAlert styles have deprecated. There are new constants currently being used in beta testing. The numeric values of the constants have remained the same but the names have changed. All the constants now all start with "NSAlertStyle." These new constants are:

- NSAlertStyleWarning (integer constant = 0)
- NSAlertStyleInformational (integer constant = 1)
- NSAlertStyleCritical (integer constant = 2)

## Alert attributes

NSAlert objects have the following attributes:

- Type: (Integer) NSWarningAlertStyle, NSInformationalAlertStyle, NSCriticalAlertStyle

- Message text: (String) The main message of the alert.

Informative text: (String) Additional information about the alert.

Response buttons: (set of NSButtons) The buttons the user can click.

Suppression checkbox: (boolean) Show or hides the suppression checkbox.

Accessory view: (NSView) Where the controls and fields specified by scripter are placed.

Icon: This is not currently implemented in the "display enhanced alert" handler.

Help: This is not currently implemented in the "display enhanced alert" handler.

## Where information appears in the Alert

NSAlerts display information in a particular order from top to bottom, except the icon which is to the left of all the other information. The order is listed below:

Message text
Informative text
Accessory view
Suppression checkbox
Response buttons

The icon is show in the top left corner of the alert: not currently implemented

The accessory view is where the: text fields, labeled fields, checkboxes, popups, radio button matrixes, path controls and rules are placed.

## User Buttons

A default button is automatically created when a NSAlert is called. The button uses the "return key" as a keyboard shortcut.

Any button with a title of "Cancel" will automatically use the escape key as a keyboard shortcut.

Any button with the title "Don't Save" and is not the first (rightmost) button will automatically use the keyboard shortcut Command-D as a keyboard shortcut.

Other key board shortcuts for buttons can be added by using the method "setKeyEquivalent" of the NSButton class.

All buttons, except the the default button, have to be added using the method "addButtonWithTitle"

The first button (the "default button") is placed at the far right of button row at the bottom of the alert. Each new button is added to the left of the previously existing button(s).

## Identifying which button was clicked

Buttons are identified by their horizontal position starting from the rightmost button. The number to identify the Nth most button from the right is equal to 999 + N, e.g. 1000 means the rightmost is being referenced, 1002 means the third button from the right edge of the dialog is being

referenced, 1005 means the sixth button from the right edge of the dialog is being referenced, and so on.

There are 3 predefined integer constants to identify the first 3 buttons:

NSAlertFirstButtonReturn = 1000, which means the first button was clicked
NSAlertSecondButtonReturn = 1001, which means the second button was clicked
NSAlertThirdButtonReturn = 1002 which means the third button was clicked

## Suppression checkbox

The "Suppression checkbox" is a property of NSAlert. When "TheAlert.showsSuppressionButton" is set to "Yes" (The variable "TheAlert" holds a pointer to the Alert data structure in memory as opposed to the data structure itself.) a check box followed by programmer specified text is displayed on the Alert. If "TheAlert.showsSuppressionButton" is set to "No" then then neither the check box nor the programmer specified text is displayed and the row of buttons moves up to fill in the place that would have been uses for the check box. By default this value is set to "No" which which does not show the check box and is associated suppression text.

The suppression checkbox allows the user to click the suppression checkbox, which will suppress the display of future alerts when the event that triggered the current alert occurs again.

A suppression checkbox has the following text displayed to the right of the text box: “Do not show this message again” This is the default text displayed. The text is actually the title of the checkbox. This text can be changed by the following expression:

```
[[alert suppressionButton] setTitle:@"My new button title."];
```

This expression references the suppressionButton of the alert and sets it to the new value.

# Chapter 3: "display enhanced alert"

Note: This chapter covers concepts necessary to using "display enhanced alert."

## "display enhanced alert" input parameters

The parameters of "display enhanced alert" are:

direct parameter (called "Message text" in NSAlert)
message (called "Informative text" in NSAlert)
as (called "Alert style" in NSAlert)
buttons (NSAlert property is called "buttons," aka "Response buttons" in Obj-C docs)
suppression (aka SuppressionButton in Obj-C docs)
giving up after (Not done by NSAlert, entirely implemented in "display enhanced alert")
acc view width (Used to create an "Accessory View" where alert items are placed)
acc view height (Used to create an "Accessory View" where alert items are placed)
acc view controls (A list of all the controls to be created by "display enhanced alert")

To see where various things will appear in a dialog see "Where information appears in the Alert" in Chapter 1.

## "display enhanced alert" output

The output consists of a list of the three items: the name of the item that dismissed the alert, the boolean state of the suppression checkbox, and a list of the values of the controls passed into "acc view controls"

First item in list: It will either be the name of the button clicked, or "Gave Up" which indicates the alert timed out from exceeding the time specified in the "giving up after" input parameter.

Second item in list: The boolean state of the suppression checkbox at the time the dialog closed.

Third item in list: A list consisting of the name of the button pressed (or 'Gave Up'), the boolean state of the suppression button, and a list of the values of the controls passed in 'acc view controls'.

"display enhanced alert" can take input from14 different handlers to create controls for the alert. Table 1 shows what those 14 handlers return.

Any data returned from these handlers that "does not" the symbol ‡ following it's description in table 1 "will" be used by the "display enhanced alert" handler to create the alert. Any data returned from these handlers that "does" have the symbol ‡ following it's description "will not" be used by "display enhanced alert" handler.

| Name of handler | Data returned by handler: listed by list item number |
|---|---|
| **create field** | 1: NSTextField: Text field that receives text<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |

| Name of handler | Data returned by handler: listed by list item number |
|---|---|
| **create top labeled field** | 1: NSTextField: Text field that receives text<br>2: NSTextField: The label<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |
| **create side labeled field** | 1: NSTextField: Text field that receives text<br>2: NSTextField: The label<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>4: class of "field left" input parameter: The width of the field ‡ |
| **create path control** | 1: NSPathControl:<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |
| **create labeled path control** | 1: NSPathControl: Holds selected path<br>2: NSTextField: The label<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |
| **create checkbox** | 1: NSButton: The checkbox<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |
| **create labeled checkbox** | 1: NSButton: The checkbox<br>2: NSTextField: The label<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>4: class of the "checkbox left" input parameter:<br>The distance between accessory view's left and checkbox ‡ |
| **create popup** | 1: NSPopUpButton: The popup<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |
| **create labeled popup** | 1: NSPopUpButton: The popup<br>2: NSTextField: The label<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>4: class of the " popup left" input parameter:<br>The distance between accessory view's left and checkbox ‡ |
| **create matrix** | 1: NSMatrix: The matrix (radio buttons)<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>3: real: distance between accessory view's left and checkbox ‡ |

| Name of handler | Data returned by handler: listed by list item number |
|---|---|
| **create labeled matrix** | 1: NSMatrix: The matrix (radio buttons)<br>2: NSTextField:<br>3: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>4: real: class of the "matrix left" input parameter:<br>The distance between accessory view's left and checkbox ‡ |
| **create label** | 1: NSTextField: The label<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡<br>3: real: The actual width ‡ |
| **create rule** | 1: NSBox: Used to draw the line on alert<br>2: class of the "bottom" input parameter:<br>Distance from top of control to bottom of accessory view ‡ |

Table 1: Types returned by handlers that create controls

‡ This returned item is not used by "display enhanced alert"

# Appendix A: Description of how "display enhanced alert" works

Note: "This Appendix covers advanced topics most readers do not need to know.

Lines 2 -8: StyleType (aka "Alert style") is set to the proper integer value.
Line 9-12: A check is performed to verify user interaction is allowed. If it isn't allowed an error is created.

Line 14: An NSRect is created using NSMakeRect ( x, y, width, height ). The "x" and "y" specify the location of a rectangle while the user supplied width and height form the actual rectangle. The origin is always specified as [0, 0]. The NSRect is used as input for initWithFrame. theAccessoryView is set to a new instance of a fully initialized NSView created from alloc()'s initWithFrame.

LineL 15: Sets up a subview where the controls and text fields specified in controlsList are placed.

Line 17: Reverses the order of the buttons in buttonsList. The button order in AppleScript is the reverse of the order in Obj-C.

Line 19: Creates new instance of NSAlert and initializes it.

Lines 22-30: Sets various setting for the NSAlert.

Line 22: Determines type of NSAlert to be created. See "Alert styles" in chapter 1.

Lines 23 & 24: Set the Message & Informative fields.
See "Where information appears in the Alert" in chapter 1.

Lines 25 & 27: Adds all the buttons specified in "buttonsList " to the alert.

Line 28: Set the checkbox suppression state of the alert to the boolean sate specified by "showSuppression"

Line 29: set the AccessoryView of the Alert to the view created by line 14

Line 30: Let the default behavior handle key event as opposed to programmer handling it.

Line 33: if giveUp > 0 then abortModal method will be set up to run after "giveUp" seconds have passed. abortModal causes control to exit the modal event loop and return.

Line 35: Invokes the method (runs the handler) "showTheAlert" on the main thread using the default mode and blocks the current thread until after "showTheAlert" has completed on the main thread. The handler "showTheAlert" sets the value of returnCode to NSModalResponse. NSModalResponse is set to the result of runModal which is inside the "showTheAlert" handler.

Line 37: If a nonzero "give up" time was specified and the returnCode from the previous line (performSelectorOnMainThread) was not cancelPreviousPerformRequestsWithTarget then there is a timer running for abortModal method from the previous line that needs to be canceled. Therefore the abortModal request made in line 33 is canceled.

Line 39: The state of the suppressionButton at the time the dialog closed is copied to the variable suppressedState.

Line 40: Obj-C returns a number representing which button was clicked. Unfortunate 1000 is returned when the first button is clicked, 1001 is returned for the second button and so on. To make it easier to work with in "display enhanced alert" 1 is added "returnCode **mod** 1000" and returnCode = 1 when button 1 is clicked.

Line 41: if buttonNumber = 0 then the alert timed out

Line 42: Since the alert timed out set buttonName to "Gave Up"

Line 44: Since buttonNumber is valid set buttonName to clicked button name

Line 47: set controlResults to {}

Line 48: start repeat loop to go through all the the controls in controlsList

Lines 49-50: if aControl's kind = NSTextField then add aControl to end of controlResults

Lines 51-52: if aControl's kind = NSPopUpButton then add selection of aControl to end of controlResults

Lines 53-54: if aControl's kind = NSButton then add aControl's state to end of controlResults

Lines 55-56: if aControl's kind = NSPathControl then add aControl's URL's path to end of controlResults

Lines 57-58: if aControl's kind = NSMatrix then add aControl's selectedCell's title to end of controlResults

Lines 59-60: if aControl's kind = NSBox then add "missing value" to end of controlResults

Lines 62: end repeat loop

```
01 on display enhanced alert mainText message theExplanation as styleType buttons buttonsList suppression
showSuppression giving up after giveUp acc view width theWidth acc view height theHeight acc view controls
controlsList
02     if styleType = critical alert then
03         set styleNum to 2
04     else if styleType = warning alert then
05         set styleNum to 0
06     else
07         set styleNum to 1
08     end if
09     set theError to current application's AEInteractWithUser(-1, missing value, missing value) -- -1 is
kAEDefaultTimeout
10     if theError is not 0 then
11         error "User interaction disallowed" number theError
12     end if
13     -- make the accessory view
14     set theAccessoryView to current application's NSView's alloc()'s initWithFrame:(current application's
NSMakeRect(0, 0, theWidth, theHeight))
15     theAccessoryView's setSubviews:controlsList
16     -- reverse buttons because they get added in reverse order cf AS
17     set buttonsList to reverse of buttonsList
18     -- create an alert
19     set theAlert to current application's NSAlert's alloc()'s init()
20     -- set up alert
21     tell theAlert
22         its setAlertStyle:styleNum
23         its setMessageText:mainText
24         its setInformativeText:theExplanation
25         repeat with anEntry in buttonsList
26             (its addButtonWithTitle:anEntry)
27         end repeat
28         its setShowsSuppressionButton:showSuppression
29         its setAccessoryView:theAccessoryView
30         its (|window|()'s setAutorecalculatesKeyViewLoop:true)
31     end tell
```

```
32    -- if giveUp value > 0, tell the app to abort any modal event loop after that time, and thus close the panel
33    if giveUp > 0 then current application's NSApp's performSelector:"abortModal" withObject:(missing value) afterDelay:giveUp inModes:{current application's NSModalPanelRunLoopMode}
34    -- show alert in modal loop on main thread
35    my performSelectorOnMainThread:"showTheAlert:" withObject:theAlert waitUntilDone:true
36    --      if a giveUp time was specified and the alert didn't timeout, cancel the pending abort request
37    if giveUp > 0 and returnCode is not current application's NSModalResponseAbort then current application's NSObject's cancelPreviousPerformRequestsWithTarget:(current application's NSApp) selector:"abortModal" object:(missing value)
38    -- get values after alert is closed
39    set suppressedState to theAlert's suppressionButton()'s state() as boolean
40    set buttonNumber to returnCode mod 1000 + 1 -- where 1 = right-most button
41    if buttonNumber = 0 then
42          set buttonName to "Gave Up"
43    else
44          set buttonName to item buttonNumber of buttonsList
45    end if
46    -- get values from controls
47    set controlResults to {}
48    repeat with aControl in controlsList
49          if (aControl's isKindOfClass:(current application's NSTextField)) as boolean then
50                set end of controlResults to aControl's stringValue() as text
51          else if (aControl's isKindOfClass:(current application's NSPopUpButton)) as boolean then
52                set end of controlResults to aControl's titleOfSelectedItem() as text
53          else if (aControl's isKindOfClass:(current application's NSButton)) as boolean then
54                set end of controlResults to aControl's state() as boolean
55          else if (aControl's isKindOfClass:(current application's NSPathControl)) as boolean then
56                set end of controlResults to aControl's |URL|()'s |path|() as text
57          else if (aControl's isKindOfClass:(current application's NSMatrix)) as boolean then
58                set end of controlResults to aControl's selectedCell()'s title() as text
59          else -- NSBox
60                set end of controlResults to missing value
61          end if
62    end repeat
63    return {buttonName, suppressedState, controlResults}
```