Contents

Abstract

This is a help file for the AppleScript mode in Alpha. This file should be located in the Help subfolder of Alpha's folder to show up automatically in the Help menu when Alpha is loaded.

1 Introduction

AppleScript mode is a mode for the text editor Alpha: it is designed to facilitate editing, compiling, decompiling, executing and processing of scripts written in the AppleScript language, with all the powerful editing facilities provided by Alpha.

Once you are in AppleScript mode, there is a new menu (designated by the AppleScript editor icon) in the menu bar giving access to all the features described below.

AppleScript mode requires the Tclapplescript extension to be present on your system. It is, for instance, distributed with the Batteries Included Tcl/Tk distribution. Note that the AppleScript Editor application does not have to be launched or even present on your computer for this mode to work. AppleScript mode relies entirely on the Tclapplescript extension which interacts directly with the MacOS Open Scripting Architecture.

The AppleScript mode is identified, in the modes pop-up menu at the right of the status bar, by the four-letters code *Scrp*.

2 Installation

If you have Apple Script Mode as a part of the distribution of Alpha, you do not have anything to do: it is installed with the whole system. If you received it as a separate package, follow the instructions below which describe two methods of installation.

2.1 Automatic installation

Open the *OPEN TO INSTALL* file. Opening this file indicates to Alpha that a new package has to be installed: the procedure is automatic. Alpha knows where to store the different elements of your AppleScript Mode package.

2.2 Manual installation

- put the *Applescript Mode* folder in the *Modes* subfolder of the *Tcl* folder which is located at the same level as your Alpha application
- put the *AppleScript Help* file in the *Help* folder located at the same level as your Alpha application. Next time you launch Alpha, you will have an *AppleScript Help* item in the Help menu to edit this file.
- put the files *AppleScript-Example* and *AppleScript-Example.tcl* in the *Examples* folder located at the same level as your Alpha application.
- if there is an *AppleScripts* folder (containing sample scripts), it should be located at the same level as Alpha itself.

- launch Alpha. You have to rebuild the package indices and the Tcl ones. *Rebuild Package Indices* is in the Config–Packages menu, and *Rebuild Tcl Indices* is in the Tcl menu.
- quit Alpha and relaunch it: that's all there is to it.

2.3 After installing

Read the AppleScript Tutorial which can be opened directly from the AppleScript menu. For reference info, read the AppleScript Help file accessed from the Help menu. Have a look at the mode specific preferences: create or open any AppleScript source file and choose *Preferences...* in the *Mode Prefs* submenu of the *Config* menu (or simply hit F12 when in AppleScript mode).

3 Description of the AppleScript menu

Here is the description of the AppleScript menu items:

New Script Open a new editing window in AppleScript mode. This does not create a file on disk until you save the window.

Decompile A Script... Brings up a dialog to select a compiled script: the script is decompiled and edited in a new window in AppleScript mode.

Run A Script... Brings up a dialog to select a compiled script and run it. The result of the excution is output in the **AppleScript Results** window.

Apple Script Flags This submenu contains various flags concerning scripts compilation and some preferences. See the AppleScript Mode Tutorial for various examples. These flags are synchronized with the corresponding mode preferences (see AppleScript mode preferences below).

Augment Context This flag supposes that you are compiling the current window as a context: if it has already been compiled and if this flag is set, then the new instructions will be added to the already existing context. Otherwise they replace what is already in the context.

Include Description Add an environment at the beginning of the source script to specify a descriptive comment which will be stored in the compiled script.

Inherit From Parent When compiling a script as a context, this flag lets you specify another context script from which it will inherit: all the properties, handlers, variables etc. (bindings in the AppleScript parlance) stored in the parent will be available to the new context.

Launch To Get Terminology This flag affects the *Open a dictionary* command. If it is set, the command will first look for dynamic terminology in the application whose dictionary you open: if there is dynamic terminology, the application has to be launched in order to provide this terminology. This flag lets you specify if you accept that the application be launched. Otherwise, AppleScript Mode will

look for static terminology which does not require the application to be launched: if dynamic terminology is available, AppleScript mode will print a warning to let you know anyway. Not all applications have dynamic terminology: it concerns mainly applications supporting plugins and extensions (Alpha8 has dynamic terminology, Alpha7 does not).

Make Context Compile the current window as a context script. This script can be used later to compile another script in this context. Note that when you compile a script with this flag set, it is immediately executed in order to make its data available: for an ordinary script, executing some action rather than defining data, this is generally not what you want.

Run In Context This flag lets you run a script in a specific context. The context can be selected by opening, with the Control key held down, the Scripts submenu in the AppleScript menu.

Compile Compile the contents of the current selection (or the entire contents of the topmost window if no region is selected). In AppleScript Mode, this means compiling as binary code and storing the result on disk. If the window does not correspond yet to a file on disk (for instance if it is a new script created using the *New Script* item), the compiled script will be written in the AppleScripts folder in Alpha's folder. Otherwise the compiled script will be in the same folder as the source file. If there is no selection, the entire contents of the current window are compiled; otherwise only the selection is compiled.

Execute This is equivalent to compiling and running the script edited in the topmost window. In this case the script is just compiled internally: it is not written on disk and the compiled script is freed from memory as soon as its execution completes. The result of the execution is sent to the **AppleScript Results** window. If there is no selection, the entire contents of the current window are executed; otherwise only the selection is executed.

Run Run the script edited in the current window. It must have been previously compiled. If it is not compiled yet or if it has been changed since the last compilation, Alpha will let you know and offer to compile or recompile the script before running it. The result is sent to the **AppleScript Results** window.

Check Syntax Check the syntax of the script edited in the current window. Any error will be reported in the **AppleScript Results** window.

Apple Script Utils This submenu contains subsidiary features.

Add Description Insert a special comment at the beginning of the edited script to give a description the script. This comment is stored in the compiled script. In AppleScript Editor, it is shown in the *Description* edit field at the top of the editing window. In Alpha, we display it in a comment with the following structure:

```
(* Description
     Text of the comment here
*)
```

This comment must be located at the beginning of the script prior to any instruction. If the *Include Description* preference is set (see Preferences below), Alpha will display the comment if one is found when editing a compiled script or it will write the comment into the compiled script at compilation time. If the preference is not set, the description comment will not be taken into account.

If the Option key is pressed, the item *Add Description* is changed to *Delete Description*. This will delete the descriptive comment from the edited script if there is one. You'll have to recompile to have it deleted from the compiled script too.

Line Continuation Insert the line continuation symbol (¬) which is not always easy to find on the keyboard, especially on international keyboards.

Open A Dictionary... Get the scripting terminology of a scriptable application. This procedure takes care of various possible situations which are checked in the following order:

- does the application accept the *Get AETE* Apple Event? If yes, it means that it has an 'scsz' resource and is able to provide its scripting terminology at runtime. This is useful for applications which accept external plug-ins which are also scriptable.
- does the application have resources of type 'aete'?

The terminology information (aka scripting dictionary) is output, in human readable form, in as many files as there are 'aete' resources reported.

Colour and Mark Dictionary Just as it says. Events, classes, keywords etc. are colored. The events and classes are marked: marks are accessible, in alphabetical order, in the Marks popup menu on the right border of the window.

Dump Mem Info Output information about the scripts and contexts which have been compiled during the current session and stored in memory.

Clear Memory Frees the memory occupied by compiled scripts and contexts. Once the memory is cleared, the scripts are not available anymore for immediate execution and have to be recompiled. This concerns only the scripts you want to run manually with the *Run* command. Scripts you run from the Scripts submenu or with the *Run A Script...* item are handled directly by AppleScript mode.

AppleScript Bindings Display the key bindings available in AppleScript Mode.

AppleScript Tutorial Open the AppleScript Mode tutorial in a shell window.

Scripts Folder This submenu contains the names of all the compiled scripts stored in the current working folder, i-e the folder designated by the *AppleScripts Folder* preference (by default it is a folder called *AppleScripts*

and located at the same level as the Alpha application itself). The submenu behaves differently depending on the modifier key which is pressed as you open it:

- if no modifier key is pressed, the first item is "Rebuild scripts list" which lets you update the menu when the contents of the AppleScripts folder have been modified. If you click on the name of a script, it will be executed.
- if the option-key is pressed, all the items are prefixed with the word *edit*: this allows you to decompile and edit the script instead of executing it. In this case the first item in the submenu becomes *Reveal scripts folder* which switches to the Finder and opens the corresponding folder.
- if the control-key is pressed, the menu lets you specify a script to be used as a context script. The selected script is marked with a bullet. It will be used if the *Run in context* flag is set in the Apple Script Flags submenu. In this case the first item in the submenu becomes *Select a context* and brings an info dialog window.
- if the shift-key is pressed, the menu lets you specify a context script to be used as a parent for the script you want to compile. The selected script is marked with a diamond. It will be used only if the flags *Make context* and *Inherit from parent* are both set in the Apple Script Flags submenu. In this case the first item in the submenu becomes *Select a parent context* and brings an info dialog window.

Mnemonic tip: control for context, shift for parent.

4 File name conventions

By default, when compiling the source of a script, the name of the compiled script is set like this:

- if it has no extension, an *.scpt* extension is added;
- if it has an extension different from *.scpt*, this extension is replaced by *.scpt*;
- if it already has an *.scpt* extension, then you are asked to give a name yourself for the compiled script.

This name is remembered by Alpha (until the moment when you quit Alpha), so that you are not asked over again each time you recompile your script.

When decompiling a script to edit its source code, the editing window is named after the name of the script with an additional *.scr* extension. This can be changed if necessary when you decide to save the source script.

Concerning terminology dictionaries, the output files are named after the name of the application followed by a *.dict* extension and a sequential numbering in case there are several Apple Event Terminology resources. If the resource has a name (and not only an ID number), this name is used.

In any case, the file names will be truncated and shortened so that they do not excess a 32 chars limit. For instance a script editing window called

```
GetUSBPrinterSharingVersionNumber.scpt
```

will be compiled under the name:

```
GetUSBPrinterSharingVersi\dots{}umber
```

5 Working directory

The *Scripts* submenu shows the contents of the working directory: this is the folder where your scripts will be stored when you compile them. This folder can be set and modified (see preferences below). The default name and location of the working folder is *AppleScripts* in the same folder as Alpha itself.

If this folder can't be found (if it has been deleted or renamed for instance), AppleScript mode will create a new one automatically when necessary.

6 Memory management

When you recompile a script, Alpha warns you if the script has already been compiled previously and no changes have been made since the last compilation. On the other hand, when you try to run the current window, Alpha warns you if the script has not yet been compiled or if it has been modified since the last compilation.

Each time a script is loaded or compiled, it is kept in memory and a token is returned to identify it for later reference. AppleScript Mode keeps track of these tokens so that you don't have to bother about them. Thanks to this internal mechanism, you don't have to recompile a script each time you want to run it. The same is true for scripts compiled as contexts.

You can use the *Dump Mem Info* to know which compiled scripts are currently stored in memory: it outputs a list indicating the token and the corresponding script file on disk. If you want to free the memory occupied by these compiled scripts, use the *Clear memory* command in the Apple Script Utils submenu: this will destroy all the script tokens (except for the global context which is never destroyed).

To know which scripts or contexts are currently in memory, open a Tcl shell and type one of the following commands:

```
AppleScript info scripts
AppleScript info contexts
```

7 The Mark pop-up menu

The marking procedure marks all the function definitions (aka handlers in AppleScripts parlance), i-e all the statements of the form:

```
on functname(params)
    -- instructions
end functname
```

The error statements (*on error* etc.) are not marked though by the procedure since they do not correspond to function definitions.

8 The Functions pop-up menu

There is no mode specific behaviour for the functions pop-up menu (top right of your editing window, with a pair of braces icon).

9 Option clicking the title bar

If you Option-Click on a the title bar, you get a list of all the compiled script files present in the current working folder. Clicking on any of them will decompile it and edit it in a new window.

10 Completions

There is a set of completion procedures in AppleScript mode. To enable them, you must activate the *elecCompletion* package (in "Config–" Preferences–" Features") and choose your completion key (default is F1).

There is a set of *electric* abbreviations. Just type one of the following abbreviations and hit the completion key to have them expanded to a complete structure:

- tell
- on
- if
- try
- repeat

For instance *tell* will result in:

```
tell application "|"
|
end tell
```

You can also type the first letters of an AppleScript keyword and hit the completion key. If it is recognized as a keyword, it will be completed. If there are different possible completions, they will be listed in the status bar: hit the completion key again and you will have a list of all the possibilities from which you can select the one you want.

11 Key bindings

A few mode specific key bindings are defined in AppleScript mode: all of them use the combination 'ctrl-a' followed by a letter. For instance, to decompile and edit a script, hit 'ctrl-a' and then the letter 'o'. The following bindings are available:

'ctrl-a b'	to display the info about bindings
'ctrl-a c'	to compile the current window or selection
'ctrl-a d'	to get a scripting dictionary
'ctrl-a e'	to execute (compile and run) the current window
'ctrl-a f'	to free compiled scripts stored in memory
'ctrl-a l'	to insert the line continuation symbol (¬)
'ctrl-a m'	to dump memory info
'ctrl-a n'	to create a new script
'ctrl-a o'	to decompile and open a script
'ctrl-a r'	to select and run a script
'ctrl-a s'	to check the syntax of the current window

12 AppleScript Mode preferences

The *Preferences...* item in the *Mode Prefs* submenu of the *Config* menu allows you to edit specific preferences defined in the AppleScript mode.

The *AppleScripts Folder* preference lets you set your working directory where your compiled scripts will be stored. You can access any script in this folder from the *Scripts* submenu as described above. The working folder can be changed at will. If it has been deleted, moved or renamed, Alpha will recreate it as necessary.

Set the *Include Description* preference if you want AppleScript mode to take care of the description comment which is stored in compiled scripts. See above the *Add Comment* menu item for a description of the syntax of this comment.

The other preferences are

- *Augment Context*
- *Inherit From Parent*
- *Launch To Get Terminology*
- *Make Context*
- *Run In Context*

They have the same meaning as the corresponding flags in the Apple Script Flags submenu: preferences and submenu flags are synchronized. Settings made in the preferences will be remembered the next time you launch Alpha and will be reflected by the submenu.

13 Version History

- 1.0.1 - 01/03/31 - Previous versions of AppleScript mode by John Sarapata <sarapata_john@jpmorgan.com>
- 2.0 - 02/03/12 - Complete rewriting of the AppleScript mode to take advantage of the Tclapplescript shared library and Alpha8.
- 2.1 - 02/06/07 - Added a preference to set the working directory.
- 2.2 - 02/09/25 - Modified ASTerminology to rely on TclAE library for parsing. Enhanced the mode to handle contexts and various compilation flags. Choice of format for tutorial. Doc updated.

14 Known problems

- There is currently no possibility of recording.

Please e-mail any problem or bug you encounter: bdesgraupes@easyconnect.fr
Goto the Alpha utilities page:
`http://webperso.easyconnect.fr/bdesgraupes/alpha.html`

15 License and Disclaimer

This software is free software and distributed under the terms of the new BSD license: Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Bernard Desgraupes nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors *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 the contributors 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.