
AMBundleAction Class Reference

Interapplication Communication



2006-10-26



Apple Inc.
© 2006 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, AppleScript, Cocoa, Mac, Mac OS, and Xcode are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

AMBundleAction Class Reference 5

Overview	5
Subclassing Notes	6
Tasks	6
Initializing the Action	6
Setting and Getting Action Properties	6
Instance Methods	7
awakeFromBundle	7
bundle	7
hasView	7
initWithDefinition:fromArchive:	8
parameters	8
setParameters:	9
view	9

Document Revision History 11

AMBundleAction Class Reference

Inherits from	AMAction : NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Automator.framework
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Automator Programming Guide
Declared in	AMBundleAction.h
Related sample code	Apply Firmware Password AutomatorHandsOn CoreRecipes UnsharpMask

Overview

Instances of the `AMBundleAction` class (or its public subclasses) manage Automator actions that are loadable bundles. Automator loads action bundles from standard locations in the file system: `/System/Library/Automator`, `/Library/Automator`, and `~/Library/Automator`.

`AMBundleAction` objects have several important properties:

- The `NSBundle` object associated with the action's physical bundle
- The action's view, which holds its user interface
- A parameters dictionary that reflects the settings in the user interface

When you create a Cocoa Automator Action project in Xcode, the project template includes a custom subclass of `AMBundleAction`. (This custom class is given the name of the project.) The sole requirement for this custom class is to provide an implementation of `runWithInput:fromAction:error:`, which is declared by the superclass `AMAction`.

Subclassing Notes

As noted in “[Class Description](#)” (page 5), the project template for Cocoa Automator actions includes partially completed header and source files for a custom subclass of `AMBundleAction`. The name of this custom class is the name of the Xcode project. To complete the implementation of this subclass, you must override `runWithInput:fromAction:error:` (declared by `AMAction`).

Another reason for subclassing `AMBundleAction` is to obtain a class that is a peer to `AMAppleScriptAction`, itself a subclass of `AMBundleAction`. For example, the `AMShellScriptAction` class is a subclass of `AMBundleAction` whose instances can control the behavior of an action through shell, Perl, and Python scripts.

Methods to Override

To subclass `AMBundleAction`, you must override the `runWithInput:fromAction:error:` to implement the task performed by the action. If you have added any instance variables, you must override the `initWithDefinition:fromArchive:` (page 8) method and the `writeToDictionary:` method of `AMAction` to work with them.

Tasks

Initializing the Action

- `awakeFromBundle` (page 7)
Sent to the receiver when all objects in its bundle have been unarchived.
- `initWithDefinition:fromArchive:` (page 8)
Initializes and returns an allocated `AMBundleAction` object.

Setting and Getting Action Properties

- `bundle` (page 7)
Returns the receiver’s bundle object.
- `hasView` (page 7)
Returns whether the receiver has a view associated with it.
- `view` (page 9)
Returns the receiver’s view object.
- `parameters` (page 8)
Returns the receiver’s parameters.
- `setParameters:` (page 9)
Sets the parameters of the receiver to *newParameters*.

Instance Methods

awakeFromBundle

Sent to the receiver when all objects in its bundle have been unarchived.

- (void)awakeFromBundle

Discussion

This method allows an action object to perform set-up tasks requiring the presence of all bundle objects, such as adding itself as an observer of notifications, dynamically establishing bindings, and dynamically setting targets and actions.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [initWithDefinition:fromArchive:](#) (page 8)

Declared In

AMBundleAction.h

bundle

Returns the receiver's bundle object.

- (NSBundle *)bundle

Discussion

Returns `nil` if no bundle has been set.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [view](#) (page 9)

Declared In

AMBundleAction.h

hasView

Returns whether the receiver has a view associated with it.

- (BOOL)hasView

Availability

Available in Mac OS X v10.4 and later.

See Also

- [view](#) (page 9)

Declared In

AMBundleAction.h

initWithDefinition:fromArchive:Initializes and returns an allocated `AMBundleAction` object.

```
- (id)initWithDefinition:(NSDictionary *)dict fromArchive:(BOOL)archived
```

Discussion

The definitions dictionary *dict* contains configuration information specific to the receiver. If *archived* is YES, the definitions are coming from an archive. You may examine the definitions dictionary to learn about specific properties and settings of the action, but some of the keys are private to Automator. You should not attempt to change the values in *dict*, but you may add custom key-value pairs to the definition dictionary by overriding the `writeToDictionary:` method declared by the superclass, `AMAction`. If at runtime you need to learn about or change the action's properties in its information property list (`Info.plist`), send the appropriate `NSDictionary` messages to the action bundle's `infoDictionary`; for example:

```
[NSDictionary *infoDict = [[self bundle] infoDictionary];
NSString *theApp = [infoDict objectForKey:@"AMApplication"];
if ([theApp isEqualToString:@"Finder"]) {
    // do something appropriate
}
```

Availability

Available in Mac OS X v10.4 and later.

See Also

- [awakeFromBundle](#) (page 7)
- [definition](#) (`AMAction`)

Declared In

AMBundleAction.h

parameters

Returns the receiver's parameters.

```
- (NSMutableDictionary *)parameters
```

Discussion

The parameters of an action reflect the choices made and values entered in the action's user interface.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setParameters:](#) (page 9)

Related Sample Code

CoreRecipes

Declared In

AMBundleAction.h

setParameters:

Sets the parameters of the receiver to *newParameters*.

```
- (void)setParameters:(NSMutableDictionary *)newParameters
```

Discussion

The parameters of an action reflect the choices made and values entered in the action's user interface. Keys in the parameters dictionary identify specific user-interface objects. If an action uses the Cocoa bindings mechanism, the parameters of an `AMBundleAction` object are automatically set. You can change the parameters wholesale with this method. Or you can get the current parameters dictionary with the [parameters](#) (page 8) and update individual parameters.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [parameters](#) (page 8)

Declared In

AMBundleAction.h

view

Returns the receiver's view object.

```
- (NSView *)view
```

Availability

Available in Mac OS X v10.4 and later.

See Also

- [hasView](#) (page 7)

- [bundle](#) (page 7)

Declared In

AMBundleAction.h

Document Revision History

This table describes the changes to *AMBundleAction Class Reference*.

Date	Notes
2006-10-26	Minor modifications for Automator changes in Mac OS X version 10.5.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History