NSScriptCommandDescription Class Reference
Contents

NSScriptCommandDescription Class Reference 3
  Overview 3
  Adopted Protocols 3
  Tasks 4
    Initializing a Script Command Description 4
    Getting Basic Information About the Command 4
    Getting Command Argument Information 4
    Getting Command Return-Type Information 5
    Creating Commands 5
  Instance Methods 5
    appleEventClassCode 5
    appleEventCode 6
    appleEventCodeForArgumentWithName: 6
    appleEventCodeForReturnType 7
    argumentNames 7
    commandClassName 8
    commandName 8
    createCommandInstance 8
    createCommandInstanceWithZone: 9
    initWithSuiteName:commandName:dictionary: 9
    isOptionalArgumentWithName: 10
    returnType 11
    suiteName 11
    typeForArgumentWithName: 11

Document Revision History 13
Inherits from | NSObject
---|---
Conforms to | NSCoding
 | NSObject (NSObject)
Framework | /System/Library/Frameworks/Foundation.framework
Availability | Available in OS X v10.0 and later.
Companion guide | Cocoa Scripting Guide
Declared in | NSScriptCommandDescription.h

**Overview**

An instance of `NSScriptCommandDescription` describes a script command that a Cocoa application supports.

A scriptable application provides scriptability information that describes the commands and objects scripters can use in scripts that target the application. An application's scripting information is collected automatically by an instance of `NSScriptSuiteRegistry`, which creates an `NSScriptCommandDescription` for each command it finds, caches these objects in memory, and installs a command handler for each command.

A script command instance stores the name, class, argument types, and return type of a command. For example, commands in AppleScript's Core suite include `clone`, `count`, `create`, `delete`, `exists`, and `move`.

The public methods of `NSScriptCommandDescription` are used primarily by Cocoa's built-in scripting support in responding to Apple events that target the application. Although you can subclass the `NSScriptCommandDescription` class, it is unlikely that you would need to do so, or to create instances of it.

**Adopted Protocols**

NSCoding

- `encodeWithCoder:`
Tasks

Initializing a Script Command Description

- initWithSuiteName:commandName:dictionary: (page 9)
  Initializes and returns a newly allocated instance of NSScriptCommandDescription.

Getting Basic Information About the Command

- appleEventClassCode (page 5)
  Returns the four-character code for the Apple event class of the receiver's command.

- appleEventCode (page 6)
  Returns the four-character code for the Apple event ID of the receiver's command.

- commandClassName (page 8)
  Returns the name of the class that will be instantiated to handle the command.

- commandName (page 8)
  Returns the name of the command.

- suiteName (page 11)
  Returns the name of the suite that contains the command described by the receiver.

Getting Command Argument Information

- appleEventCodeForArgumentWithName: (page 6)
  Returns the Apple event code for the specified command argument of the receiver.

- argumentNames (page 7)
  Returns the names (or keys) for all arguments of the receiver's command.

- isOptionalArgumentWithName: (page 10)
  Returns a Boolean value that indicates whether the command argument identified by the specified argument key is an optional argument.

- typeForArgumentWithName: (page 11)
  Returns the type of the command argument identified by the specified key.
Getting Command Return-Type Information

- `appleEventCodeForReturnType` (page 7)
  Returns the Apple event code that identifies the command’s return type.

- `returnType` (page 11)
  Returns the return type of the command.

Creating Commands

- `createCommandInstance` (page 8)
  Creates and returns an instance of the command object described by the receiver.

- `createCommandInstanceWithZone:` (page 9)
  Creates and returns an instance of the command object described by the receiver in the specified memory zone.

Instance Methods

`appleEventClassCode`

*Returns the four-character code for the Apple event class of the receiver’s command.*

- `(FourCharCode)appleEventClassCode`

Return Value
The Apple event code associated with the receiver’s command. This is the primary code used to identify the command in Apple events.

Discussion
In an Apple event that specifies a script command, two four character codes—the event class and event ID—together identify the command. You use this method to obtain the event class. You use `appleEventCode` (page 6) to obtain the event ID.

For example, commands in AppleScript’s Core suite, such as `clone`, `count`, and `create`, have an event class code of 'core'. This code and the event ID code returned by `appleEventCode` together specify the necessary information for identifying and dispatching an Apple event.

Availability
Available in OS X v10.0 and later.
appleEventCode

Returns the four-character code for the Apple event ID of the receiver’s command.

- (FourCharCode)appleEventCode

Return Value
The code for the event ID of the receiver’s command.

Discussion
This value of the event ID returned by this method, together with the event class code returned by appleEventClassCode (page 5), specifies the necessary information for identifying and dispatching an Apple event.

Availability
Available in OS X v10.0 and later.

See Also
– appleEventCodeForArgumentWithName: (page 6)
– appleEventCodeForReturnType (page 7)

appleEventCodeForArgumentWithName:

Returns the Apple event code for the specified command argument of the receiver.

- (FourCharCode)appleEventCodeForArgumentWithName:(NSString *)argumentName

Parameters
argumentName
The argument name (used as a key) for which to obtain the corresponding Apple event code.

Return Value
The code for the specified argument.

Availability
Available in OS X v10.0 and later.
See Also
− argumentNames (page 7)

Declared in
NSScriptCommandDescription.h

appleEventCodeForReturnType

Returns the Apple event code that identifies the command’s return type.

− (FourCharCode)appleEventCodeForReturnType

Return Value
The event code for the command’s return type.

Availability
Available in OS X v10.0 and later.

See Also
− appleEventCodeForArgumentWithName: (page 6)
− returnType (page 11)

Declared in
NSScriptCommandDescription.h

argumentNames

Returns the names (or keys) for all arguments of the receiver’s command.

− (NSArray *)argumentNames

Return Value
The array of argument names. If there are no arguments for the command, returns an empty array.

Availability
Available in OS X v10.0 and later.

Declared in
NSScriptCommandDescription.h
**commandClassName**

*Returns the name of the class that will be instantiated to handle the command.*

- (NSString *)commandClassName

**Return Value**
The Objective-C class name (for example, "NSGetCommand"). This is always NSScriptCommand or a subclass.

**Availability**
Available in OS X v10.0 and later.

**See Also**
– commandName (page 8)

**Declared in**
NSScriptCommandDescription.h

---

**commandName**

*Returns the name of the command.*

- (NSString *)commandName

**Return Value**
The command name as it appears in the application's scriptability information; may be different from what is displayed to the scripter.

**Availability**
Available in OS X v10.0 and later.

**See Also**
– commandClassName (page 8)

**Declared in**
NSScriptCommandDescription.h

---

**createCommandInstance**

*Creates and returns an instance of the command object described by the receiver.*

- (NSScriptCommand *)createCommandInstance
Return Value
The command object, instantiated from NSScriptCommand or a subclass.

Availability
Available in OS X v10.0 and later.

Declared in
NSScriptCommandDescription.h

createCommandInstanceWithZone:

Creates and returns an instance of the command object described by the receiver in the specified memory zone.

- (NSScriptCommand *)createCommandInstanceWithZone:(NSZone *)zone

Parameters
zone
The memory zone from which to allocate the command.

Return Value
The command object, instantiated from NSScriptCommand or a subclass.

Availability
Available in OS X v10.0 and later.

Declared in
NSScriptCommandDescription.h

initWithSuiteName:commandName:dictionary:

Initializes and returns a newly allocated instance of NSScriptCommandDescription.

- (id)initWithSuiteName:(NSString *)suiteName commandName:(NSString *)commandName dictionary:(NSDictionary *)commandDeclaration

Parameters
suiteName
The name of the suite (in the application's scriptability information) that the command belongs to. For example, "AppName Suite".

commandName
The name of the script command that this instance describes.
commandDeclaration

A command declaration dictionary of the sort that is valid in script suite property list files. This dictionary provides information about the command such as its argument names and types and return type (if any).

Return Value
The initialized command description instance. Returns nil if the event constant or class name for the command description is missing; also returns nil if the return type or argument values are of the wrong type.

Discussion
This method registers self with the application's global instance of NSScriptSuiteRegistry and also registers all command arguments with the registry.

Availability
Available in OS X v10.0 and later.

Declared in
NSScriptCommandDescription.h

isOptionalArgumentWithName:

Returns a Boolean value that indicates whether the command argument identified by the specified argument key is an optional argument.

- (BOOL)isOptionalArgumentWithName:(NSString *)argumentName

Parameters
argumentName
Argument name (used as a key) that identifies the command argument to examine.

Return Value
YES if the specified argument exists and is optional; otherwise, NO.

Availability
Available in OS X v10.0 and later.

See Also
- argumentNames (page 7)

Declared in
NSScriptCommandDescription.h
**returnType**

*Returns the return type of the command.*

- (NSString *)returnType

**Return Value**
The receiver's command return type; for example, "NSNumber" or "NSDictionary".

**Availability**
Available in OS X v10.0 and later.

**See Also**
- appleEventCodeForReturnType (page 7)

**Declared in**
NSScriptCommandDescription.h

**suiteName**

*Returns the name of the suite that contains the command described by the receiver.*

- (NSString *)suiteName

**Return Value**
The receiver's suite name. Within an application's scriptability information, named suites contain related sets of information.

**Availability**
Available in OS X v10.0 and later.

**See Also**
- appleEventCode (page 6)

**Declared in**
NSScriptCommandDescription.h

**typeForArgumentWithName:**

*Returns the type of the command argument identified by the specified key.*

- (NSString *)typeForArgumentWithName:(NSString *)argumentName
Parameters
argumentName
   Argument name (used as a key) that identifies the command argument to examine.

Return Value
The type of the specified command argument. Returns nil if there is no such argument.

Availability
Available in OS X v10.0 and later.

Declared in
NSScriptCommandDescription.h
Document Revision History

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-04-10</td>
<td>Revised parameter descriptions to conform to reference consistency guidelines.</td>
</tr>
<tr>
<td>2006-05-23</td>
<td>First publication of this content as a separate document.</td>
</tr>
</tbody>
</table>
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.

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.

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

Apple, the Apple logo, AppleScript, Cocoa, Mac,
Objective-C, and OS X are trademarks of Apple
Inc., registered in the U.S. and other countries.

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.