
WebPolicyDelegate Protocol Reference

[Cocoa > User Experience](#)



2009-04-08



Apple Inc.
© 2009 Apple 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, Cocoa, Mac, Mac OS, Objective-C, and Safari are trademarks of Apple Inc., registered in the United States and other countries.

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

WebPolicyDelegate Protocol Reference 5

Overview 5

Tasks 5

 Making Content Decisions 5

 Making Navigation Decisions 6

 Making New Window Decisions 6

 Handling Errors 6

Instance Methods 6

 webView:decidePolicyForMIMEType:request:frame:decisionListener: 6

 webView:decidePolicyForNavigationAction:request:frame:decisionListener: 7

 webView:decidePolicyForNewWindowAction:request:newFrameName:decisionListener: 8

 webView:unableToImplementPolicyWithError:frame: 8

Constants 9

 Action Dictionary Keys 9

 Navigation Type Values 10

Document Revision History 11

Index 13

WebPolicyDelegate Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/WebKit.framework
Availability	Available in Mac OS X v10.2 with Safari 1.0 and later. Available in Mac OS X v10.2.7 and later.
Companion guide	WebKit Objective-C Programming Guide
Declared in	WebPolicyDelegate.h

Overview

The `WebPolicyDelegate` informal protocol works with the `WebPolicyDecisionListener` protocol to modify the policy decisions that the `WebView` class makes when handling URLs or the data objects they represent. The methods in this protocol are typically invoked in the following order.

1. The `webView:decidePolicyForNewWindowAction:request:newFrameName:decisionListener:` (page 8) method is invoked once for every load.
2. The `webView:decidePolicyForNavigationAction:request:frame:decisionListener:` (page 7) method may be invoked zero or more times after a load started. This method is invoked every time a server redirect is encountered unless blocked by an earlier policy decision.
3. The `webView:decidePolicyForMIMETYPE:request:frame:decisionListener:` (page 6) method is invoked after the MIME type of the content is known unless this method is blocked by an earlier policy decision.
4. The `webView:unableToImplementPolicyWithError:frame:` (page 8) method is invoked when an error occurs implementing a policy decision.

Tasks

Making Content Decisions

- `webView:decidePolicyForMIMETYPE:request:frame:decisionListener:` (page 6)
Decides whether to display content with a given MIME type.

Making Navigation Decisions

- [webView:decidePolicyForNavigationAction:request:frame:decisionListener:](#) (page 7)
Routes a navigation action internally or to an external viewer.

Making New Window Decisions

- [webView:decidePolicyForNewWindowAction:request:newFrameName:decisionListener:](#) (page 8)
Decides whether to allow a targeted navigation event, such as opening a link in a new window.

Handling Errors

- [webView:unableToImplementPolicyWithError:frame:](#) (page 8)
Handles or drops events that were rejected by a policy maker.

Instance Methods

webView:decidePolicyForMIMETYPE:request:frame:decisionListener:

Decides whether to display content with a given MIME type.

```
- (void)webView:(WebView *)webView decidePolicyForMIMETYPE:(NSString *)type
  request:(NSURLRequest *)request frame:(WebFrame *)frame decisionListener:(id
  < WebPolicyDecisionListener > )listener
```

Parameters

webView

The associated web view.

type

The MIME type of the content.

request

The request to load the content.

frame

The frame for displaying the content.

listener

The object that receives the policy decision.

Discussion

This method is invoked during the process of loading content for *request* after the `webView:didStartProvisionalLoadForFrame:` method in the `WebFrameLoadDelegate` informal protocol is called by the `WebView` object. The web view implements a policy decision by sending one of the `WebPolicyDecisionListener` protocol messages to *listener*.

If you do not implement this method, the default behavior is used. The listener is told to ignore the MIME type unless `webView` specifies it can handle the type in its `canShowMIMETYPE:` method.

In some rare cases, multiple responses may be received for a single resource. This happens in the case of multipart/x-mixed-replace, also known as a “server push.” In this case, this method will be invoked multiple times.

Availability

Available in Mac OS X v10.2 with Safari 1.0 and later.

Available in Mac OS X v10.2.7 and later.

Declared In

WebPolicyDelegate.h

webView:decidePolicyForNavigationAction:request:frame:decisionListener:

Routes a navigation action internally or to an external viewer.

```
- (void)webView:(WebView *)webView decidePolicyForNavigationAction:(NSDictionary *)actionInformation request:(NSURLRequest *)request frame:(WebFrame *)frame decisionListener:(id < WebPolicyDecisionListener >)listener
```

Parameters

webView

The `WebView` object for which this object is the policy delegate.

actionInformation

A description of the action that triggered the navigation request. The possible key-value pairs in this dictionary are defined in “[Action Dictionary Keys](#)” (page 9).

request

The request for which the navigation is made.

frame

The `WebFrame` object in which the action occurred.

listener

The `WebPolicyDecisionListener` object that receives the policy decision.

Discussion

This method is invoked when a navigation decision needs to be made. The web view implements a policy decision by sending one of the `WebPolicyDecisionListener` protocol messages to *listener*. This method is invoked whenever a server redirect is encountered, and before loading starts.

If you do not implement this method, the default behavior is used. The listener handles the navigation internally if the request is for an error page or if the `canHandleRequest:` method of the `NSURLConnection` class returns YES when passed *request*. Otherwise, the listener ignores the navigation, and it is handled externally.

Availability

Available in Mac OS X v10.2 with Safari 1.0 and later.

Available in Mac OS X v10.2.7 and later.

Declared In

WebPolicyDelegate.h

webView:decidePolicyForNewWindowAction:request:newFrameName:decisionListener:

Decides whether to allow a targeted navigation event, such as opening a link in a new window.

```
- (void)webView:(WebView *)webView decidePolicyForNewWindowAction:(NSDictionary *)actionInformation request:(NSURLRequest *)request newFrameName:(NSString *)frameName decisionListener:(id < WebPolicyDecisionListener >)listener
```

Parameters

webView

The `WebView` object for which this object is the policy delegate.

actionInformation

A description of the action that triggered the navigation request. The possible key-value pairs in this dictionary are defined in [“Action Dictionary Keys”](#) (page 9).

request

The request for which the new window action is performed.

frameName

The name of the new frame that contains the content returned from the request.

listener

The `WebPolicyDecisionListener` object that receives the policy decision.

Discussion

This method is invoked when a targeted navigation decision needs to be made. A targeted navigation typically opens a new window to display content. The receiver implements a policy decision by sending one of the `WebPolicyDecisionListener` protocol messages to *listener*. This method allows delegates to modify the behavior of targeted links which normally open a new window. Delegates might do something else, such as download or present the content in a special way. If this method sends *use* to *listener* then the new window will be opened, and

[webView:decidePolicyForNavigationAction:request:frame:decisionListener:](#) (page 7) will be invoked with a `WebNavigationTypeOther` (page ?) as the value for the `WebActionNavigationTypeKey` (page ?) key in the action dictionary.

The default behavior sends *use* to *listener*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 and later.

Available in Mac OS X v10.2.7 and later.

Declared In

`WebPolicyDelegate.h`

webView:unableToImplementPolicyWithError:frame:

Handles or drops events that were rejected by a policy maker.

```
- (void)webView:(WebView *)webView unableToImplementPolicyWithError:(NSError *)error frame:(WebFrame *)frame
```

Parameters

webView

The `WebView` object for which this object is the policy delegate.

error

The error that occurred.

frame

The frame in which the error occurred.

Discussion

Delegates might implement this method to display or log an error message. If you do not implement this method, no action is taken.

Availability

Available in Mac OS X v10.2 with Safari 1.0 and later.

Available in Mac OS X v10.2.7 and later.

Declared In

WebPolicyDelegate.h

Constants

Action Dictionary Keys

Keys that might appear in a dictionary passed as the `actionInformation` parameter to the [webView:decidePolicyForNavigationAction:request:frame:decisionListener:](#) (page 7) and [webView:decidePolicyForNewWindowAction:request:newFrameName:decisionListener:](#) (page 8) methods.

WebActionNavigationTypeKey

WebActionElementKey

WebActionButtonKey

WebActionModifierFlagsKey

WebActionOriginalURLKey

Constants

WebActionNavigationTypeKey

The navigation type of the action. Can be any of the values defined in “Navigation Type Values” below.

Available in Mac OS X v10.2 and later.

Declared in WebPolicyDelegate.h.

WebActionElementKey

A dictionary containing element information. See the “Constants” section of *WebView Class Reference* for a description of the key-value pairs in this dictionary.

Available in Mac OS X v10.2 and later.

Declared in WebPolicyDelegate.h.

WebActionButtonKey

The event type that occurred.

Available in Mac OS X v10.2 and later.

Declared in WebPolicyDelegate.h.

WebActionModifierFlagsKey

An unsigned number that indicates the modifier flag.

Available in Mac OS X v10.2 and later.

Declared in WebPolicyDelegate.h.

WebActionOriginalURLKey

The URL that initiated the action.

Available in Mac OS X v10.2 and later.

Declared in WebPolicyDelegate.h.

Navigation Type Values

These are the possible values for the `WebActionNavigationTypeKey` key that appears in an action dictionary.

WebNavigationTypeLinkClicked

WebNavigationTypeFormSubmitted

WebNavigationTypeBackForward

WebNavigationTypeReload

WebNavigationTypeFormResubmitted

WebNavigationTypeOther

Constants

WebNavigationTypeLinkClicked

A link (an href) was clicked.

WebNavigationTypeFormSubmitted

A form was submitted.

WebNavigationTypeBackForward

The user clicked back or forward button.

WebNavigationTypeReload

The user hit the reload button.

WebNavigationTypeFormResubmitted

A form was resubmitted (through a back, forward or reload action).

WebNavigationTypeOther

Navigation is taking place for some other reason.

Document Revision History

This table describes the changes to *WebPolicyDelegate Protocol Reference*.

Date	Notes
2009-04-08	Added separate parameter and return value descriptions.
2008-10-15	Minor edits throughout.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

Action Dictionary Keys [9](#)

N

Navigation Type Values [10](#)

W

WebActionButtonKey **constant** [9](#)
WebActionElementKey **constant** [9](#)
WebActionModifierFlagsKey **constant** [10](#)
WebActionNavigationTypeKey **constant** [9](#)
WebActionOriginalURLKey **constant** [10](#)
WebNavigationTypeBackForward **constant** [10](#)
WebNavigationTypeFormResubmitted **constant** [10](#)
WebNavigationTypeFormSubmitted **constant** [10](#)
WebNavigationTypeLinkClicked **constant** [10](#)
WebNavigationTypeOther **constant** [10](#)
WebNavigationTypeReload **constant** [10](#)
webView:decidePolicyForMIMEType:request:frame:
 decisionListener: **protocol instance method** [6](#)
webView:decidePolicyForNavigationAction:request:
 frame:decisionListener: **protocol instance
method** [7](#)
webView:decidePolicyForNewWindowAction:request:
 newFrameName:decisionListener: **protocol
instance method** [8](#)
webView:unableToImplementPolicyWithError:frame:
 protocol instance method [8](#)