
NSURLDownload Class Reference

Networking, Internet, & Web: Protocol Streams



2009-08-12



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, 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

NSURLDownload Class Reference 5

Overview	5
Tasks	6
Creating a Download Instance	6
Resuming Partial Downloads	6
Canceling a Download	7
Getting Download Properties	7
Setting the Destination Path	7
Download progress	7
Class Methods	8
canResumeDownloadDecodedWithEncodingMIMEType:	8
Instance Methods	8
cancel	8
deletesFileUponFailure	9
initWithRequest:delegate:	9
initWithResumeData:delegate:path:	10
request	10
resumeData	10
setDeletesFileUponFailure:	11
setDestination:allowOverwrite:	11
Delegate Methods	12
download:canAuthenticateAgainstProtectionSpace:	12
download:decideDestinationWithSuggestedFilename:	13
download:didCancelAuthenticationChallenge:	13
download:didCreateDestination:	14
download:didFailWithError:	14
download:didReceiveAuthenticationChallenge:	14
download:didReceiveDataOfLength:	15
download:didReceiveResponse:	16
download:shouldDecodeSourceDataOfMIMEType:	16
download:willResumeWithResponse:fromByte:	17
download:willSendRequest:redirectResponse:	17
downloadDidBegin:	18
downloadDidFinish:	18
downloadShouldUseCredentialStorage:	19

Document Revision History 21

Index 23

NSURLDownload Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.2 with Safari 1.0 installed. Available in Mac OS X v10.2.7 and later.
Companion guide	URL Loading System
Declared in	NSURLDownload.h
Related sample code	QuickLookDownloader

Overview

NSURLDownload downloads a request asynchronously and saves the data to a file. The interface for NSURLDownload is sparse, providing methods to initialize a download, set the destination path and cancel loading the request.

NSURLDownload's delegate methods allow an object to receive informational callbacks about the asynchronous load of the URL request. Other delegate methods provide facilities that allow the delegate to customize the process of performing an asynchronous URL load.

Note that these delegate methods will be called on the thread that started the asynchronous load operation for the associated NSURLDownload object.

- A [downloadDidBegin:](#) (page 18) message will be sent to the delegate immediately upon starting the download.
- Zero or more [download:willSendRequest:redirectResponse:](#) (page 17) messages will be sent to the delegate before any further messages are sent if it is determined that the download must redirect to a new location. The delegate can allow the redirect, modify the destination or deny the redirect.
- Zero or more [download:didReceiveAuthenticationChallenge:](#) (page 14) messages will be sent to the delegate if it is necessary to authenticate in order to download the request and NSURLDownload does not already have authenticated credentials.
- Zero or more [download:didCancelAuthenticationChallenge:](#) (page 13) messages will be sent to the delegate if NSURLDownload cancels the authentication challenge due to encountering a protocol implementation error.

- Zero or more `download:didReceiveResponse:` (page 16) messages will be sent to the delegate before receiving a `download:didReceiveDataOfLength:` (page 15) message. The only case where `download:didReceiveResponse:` is not sent to a delegate is when the protocol implementation encounters an error before a response could be created.
- Zero or more `download:didReceiveDataOfLength:` (page 15) messages will be sent before `downloadDidFinish:` (page 18) or `download:didFailWithError:` (page 14) is sent to the delegate.
- Zero or one `download:decideDestinationWithSuggestedFilename:` (page 13) will be sent to the delegate when sufficient information has been received to determine the suggested filename for the downloaded file. The delegate will not receive this message if `setDestination:allowOverwrite:` (page 11) has already been sent to the NSURLOnload instance.
- A `download:didCreateDestination:` (page 14) message will be sent to the delegate when the NSURLOnload instance creates the file on disk.
- If NSURLOnload determines that the downloaded file is in a format that it is able to decode (MacBinary, Binhex or gzip), the delegate will receive a `download:shouldDecodeSourceDataOfMIMEType:` (page 16). The delegate should return YES to decode the data, NO otherwise.
- Unless an NSURLOnload instance receives a `cancel` (page 8) message, the delegate will receive one and only one `downloadDidFinish:` (page 18) or `download:didFailWithError:` (page 14) message, but never both. In addition, once either of messages are sent, the delegate will receive no further messages for the given NSURLOnload.

Tasks

Creating a Download Instance

- `initWithRequest:delegate:` (page 9)
Returns an initialized URL download for a URL request and begins to download the data for the request.

Resuming Partial Downloads

- + `canResumeDownloadDecodedWithEncodingMIMEType:` (page 8)
Returns whether a URL download object can resume a download that was decoded with the specified MIME type.
- `initWithResumeData:delegate:path:` (page 10)
Returns an initialized NSURLOnload object that will resume downloading the specified data to the specified file and begins the download.
- `resumeData` (page 10)
Returns the resume data for a download that is not yet complete.
- `setDeletesFileUponFailure:` (page 11)
Specifies whether the receiver deletes the partially downloaded file when a download stops prematurely.
- `deletesFileUponFailure` (page 9)
Returns whether the receiver deletes partially downloaded files when a download stops prematurely.

Canceling a Download

- [cancel](#) (page 8)
Cancels the receiver's download and deletes the downloaded file.

Getting Download Properties

- [request](#) (page 10)
Returns the request that initiated the receiver's download.

Setting the Destination Path

- [setDestination:allowOverwrite:](#) (page 11)
Sets the destination path of the downloaded file.

Download progress

- [download:canAuthenticateAgainstProtectionSpace:](#) (page 12) *delegate method*
Sent to determine whether the delegate is able to respond to a protection space's form of authentication.
- [download:decideDestinationWithSuggestedFilename:](#) (page 13) *delegate method*
The delegate receives this message when *download* has determined a suggested filename for the downloaded file.
- [download:didCancelAuthenticationChallenge:](#) (page 13) *delegate method*
Sent if an authentication challenge is canceled due to the protocol implementation encountering an error.
- [download:didCreateDestination:](#) (page 14) *delegate method*
Sent when the destination file is created.
- [download:didFailWithError:](#) (page 14) *delegate method*
Sent if the download fails or if an I/O error occurs when the file is written to disk.
- [download:didReceiveAuthenticationChallenge:](#) (page 14) *delegate method*
Sent when the URL download must authenticate a challenge in order to download the request.
- [download:didReceiveDataOfLength:](#) (page 15) *delegate method*
Sent as a download object receives data incrementally.
- [download:didReceiveResponse:](#) (page 16) *delegate method*
Sent when a download object has received sufficient load data to construct the `NSURLResponse` object for the download.
- [download:shouldDecodeSourceDataOfMIMEType:](#) (page 16) *delegate method*
Sent when a download object determines that the downloaded file is encoded to inquire whether the file should be automatically decoded.
- [downloadShouldUseCredentialStorage:](#) (page 19) *delegate method*
Sent to determine whether the URL loader should consult the credential storage to authenticate the download.

- `download:willSendRequest:redirectResponse:` (page 17) *delegate method*
Sent when the download object determines that it must change URLs in order to continue loading a request.
- `downloadDidBegin:` (page 18) *delegate method*
Sent immediately after a download object begins a download.
- `downloadDidFinish:` (page 18) *delegate method*
Sent when a download object has completed downloading successfully and has written its results to disk.
- `download:willResumeWithResponse:fromByte:` (page 17) *delegate method*
Sent when a download object has received a response from the server after attempting to resume a download.

Class Methods

canResumeDownloadDecodedWithEncodingMIMEType:

Returns whether a URL download object can resume a download that was decoded with the specified MIME type.

```
+ (BOOL)canResumeDownloadDecodedWithEncodingMIMEType:(NSString *)MIMEType
```

Parameters

MIMEType

The MIME type the caller wants to know about.

Return Value

YES if the URL download object can resume a download that was decoded with the specified MIME type, NO otherwise.

Discussion

NSURLOnload cannot resume a download that was partially decoded in the gzip format.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSURLOnload.h

Instance Methods

cancel

Cancels the receiver's download and deletes the downloaded file.

```
- (void)cancel
```

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLDownload.h

deletesFileUponFailure

Returns whether the receiver deletes partially downloaded files when a download stops prematurely.

- (BOOL)deletesFileUponFailure

Return Value

YES if partially downloaded files should be deleted when a download stops prematurely, NO otherwise. The default is YES.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setDeletesFileUponFailure:](#) (page 11)

Declared In

NSURLDownload.h

initWithRequest:delegate:

Returns an initialized URL download for a URL request and begins to download the data for the request.

- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate

Parameters

request

The URL request to download. The *request* object is deep-copied as part of the initialization process. Changes made to *request* after this method returns do not affect the request that is used for the loading process.

delegate

The delegate for the download. This object will receive delegate messages as the download progresses. Delegate messages will be sent on the thread which calls this method. For the download to work correctly the calling thread's run loop must be operating in the default run loop mode.

Return Value

An initialized NSURLDownload object for *request*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLDownload.h

initWithResumeData:delegate:path:

Returns an initialized NSURLOnload object that will resume downloading the specified data to the specified file and begins the download.

```
- (id)initWithResumeData:(NSData *)resumeData delegate:(id)delegate path:(NSString *)path
```

Parameters

resumeData

Specifies the data to resume downloading.

delegate

The delegate for the download. This object will receive delegate messages as the download progresses. Delegate messages will be sent on the thread which calls this method. For the download to work correctly the calling thread's run loop must be operating in the default run loop mode.

path

The location for the downloaded data.

Return Value

An initialized NSURLOnload object.

Availability

Available in Mac OS X v10.4 and later.

See Also

[resumeData](#) (page 10)

Declared In

NSURLOnload.h

request

Returns the request that initiated the receiver's download.

```
- (NSURLRequest *)request
```

Return Value

The URL request that initiated the receiver's download.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

resumeData

Returns the resume data for a download that is not yet complete.

```
- (NSData *)resumeData
```

Return Value

The resume data for a download that is not yet complete. This data represents the necessary state information that an `NSURLOnload` object needs to resume a download. The resume data can later be used when initializing a download with `initWithResumeData:delegate:path:` (page 10). Returns `nil` if the download is not able to be resumed.

Discussion

Resume data will only be returned if the protocol of the download as well as the server support resuming. In order to later resume a download you must call `setDeletesFileUponFailure:` (page 11) passing `NO` so the partially downloaded data is not deleted when the initial connection is lost or canceled.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSURLOnload.h`

setDeletesFileUponFailure:

Specifies whether the receiver deletes the partially downloaded file when a download stops prematurely.

- (void)setDeletesFileUponFailure:(BOOL)deletesFileUponFailure

Parameters

deletesFileUponFailure

YES if partially downloaded files should be deleted when a download stops prematurely, NO otherwise. The default is YES.

Discussion

To allow the download to be resumed in case the download ends prematurely you should call this method as soon as possible after starting the download.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [deletesFileUponFailure](#) (page 9)

Declared In

`NSURLOnload.h`

setDestination:allowOverwrite:

Sets the destination path of the downloaded file.

- (void)setDestination:(NSString *)path allowOverwrite:(BOOL)allowOverwrite

Parameters

path

The path for the downloaded file.

allowOverwrite

YES if an existing file at *path* can be replaced, NO otherwise.

Discussion

If *allowOverwrite* is NO and a file already exists at *path*, a unique filename will be created for the downloaded file by appending a number to the filename. The delegate can implement [download:didCreateDestination:](#) (page 14) to determine the filename used when the file is written to disk.

Special Considerations

An NSURLOnload instance ignores multiple calls to this method.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

- [download:decideDestinationWithSuggestedFilename:](#) (page 13)
- [download:didCreateDestination:](#) (page 14)

Related Sample Code

QuickLookDownloader

Declared In

NSURLOnload.h

Delegate Methods

download:canAuthenticateAgainstProtectionSpace:

Sent to determine whether the delegate is able to respond to a protection space's form of authentication.

```
- (BOOL)download:(NSURLOnload *)download
    canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace
```

Parameters

download

The download sending the message.

protectionSpace

The protection space that generates an authentication challenge.

Discussion

This method is called before [download:didReceiveAuthenticationChallenge:](#) (page 14), allowing the delegate to inspect a protection space before attempting to authenticate against it. By returning YES, the delegate indicates that it can handle the form of authentication, which it does in the subsequent call to [download:didReceiveAuthenticationChallenge:](#) (page 14). Not implementing this method is the same as returning NO, in which case default authentication handling is used.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSURLOnload.h

download:decideDestinationWithSuggestedFilename:

The delegate receives this message when *download* has determined a suggested filename for the downloaded file.

```
- (void)download:(NSURLDownload *)download
    decideDestinationWithSuggestedFilename:(NSString *)filename
```

Parameters

download

The URL download object sending the message.

filename

The suggested filename for the download.

Discussion

The suggested filename is either derived from the last path component of the URL and the MIME type or, if the download was encoded, from the encoding. If the delegate wishes to modify the path, it should send [setDestination:allowOverwrite:](#) (page 11) to *download*.

Special Considerations

The delegate will not receive this message if [setDestination:allowOverwrite:](#) has already been called for the download.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:didCancelAuthenticationChallenge:

Sent if an authentication challenge is canceled due to the protocol implementation encountering an error.

```
- (void)download:(NSURLDownload *)download
    didCancelAuthenticationChallenge:(NSURLOnAuthenticationChallenge *)challenge
```

Parameters

download

The URL download object sending the message.

challenge

The authentication challenge that caused the download object to cancel the download.

Discussion

If the delegate receives this message the download will fail and the delegate will receive a [download:didFailWithError:](#) (page 14) message.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:didCreateDestination:

Sent when the destination file is created.

```
- (void)download:(NSURLOnload *)download didCreateDestination:(NSString *)path
```

Parameters

download

The URL download object sending the message.

path

The path to the destination file.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:didFailWithError:

Sent if the download fails or if an I/O error occurs when the file is written to disk.

```
- (void)download:(NSURLOnload *)download didFailWithError:(NSError *)error
```

Parameters

download

The URL download object sending the message.

error

The error that caused the failure of the download.

Discussion

Any partially downloaded file will be deleted.

Special Considerations

Once the delegate receives this message, it will receive no further messages for *download*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:didReceiveAuthenticationChallenge:

Sent when the URL download must authenticate a challenge in order to download the request.

```
- (void)download:(NSURLOnload *)download
  didReceiveAuthenticationChallenge:(NSURLOnloadAuthenticationChallenge *)challenge
```

Parameters*download*

The URL download object sending the message.

challenge

The URL authentication challenge that must be authenticated in order to download the request.

Discussion

This method gives the delegate the opportunity to determine the course of action taken for the challenge: provide credentials, continue without providing credentials or cancel the authentication challenge and the download.

The delegate can determine the number of previous authentication challenges by sending the message `previousFailureCount` to *challenge*.

If the previous failure count is 0 and the value returned by `proposedCredential` is `nil`, the delegate can create a new `NSURLCredential` object, providing information specific to the type of credential, and send a `useCredential:forAuthenticationChallenge: message` to `[challenge sender]`, passing the credential and *challenge* as parameters. If `proposedCredential` is not `nil`, the value is a credential from the URL or the shared credential storage that can be provided to the user as feedback.

The delegate may decide to abandon further attempts at authentication at any time by sending `[challenge sender] a continueWithoutCredentialForAuthenticationChallenge: or a cancelAuthenticationChallenge: message`. The specific action is implementation dependent.

If the delegate implements this method, the download will suspend until `[challenge sender]` is sent one of the following messages: `useCredential:forAuthenticationChallenge:`, `continueWithoutCredentialForAuthenticationChallenge:` or `cancelAuthenticationChallenge:`.

If the delegate does not implement this method the default implementation is used. If a valid credential for the request is provided as part of the URL, or is available from the `NSURLCredentialStorage` the `[challenge sender]` is sent a `useCredential:forAuthenticationChallenge:` with the credential. If the challenge has no credential or the credentials fail to authorize access, then `continueWithoutCredentialForAuthenticationChallenge:` is sent to `[challenge sender]` instead.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

`NSURLOnload.h`

download:didReceiveDataOfLength:

Sent as a download object receives data incrementally.

```
- (void)download:(NSURLOnload *)download didReceiveDataOfLength:(NSUInteger)length
```

Parameters*download*

The URL download object sending the message.

length

The amount of data received in this increment of the download, measured in bytes.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:didReceiveResponse:

Sent when a download object has received sufficient load data to construct the NSURLResponse object for the download.

```
- (void)download:(NSURLOnload *)download didReceiveResponse:(NSURLResponse *)response
```

Parameters

download

The URL download object sending the message.

response

The URL response object received as part of the download. *response* is immutable and will not be modified after this method is called.

Discussion

In some rare cases, multiple responses may be received for a single download. In this case, the client should assume that each new response resets the download progress to 0 and should check the new response for the expected content length.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:shouldDecodeSourceDataOfMIMEType:

Sent when a download object determines that the downloaded file is encoded to inquire whether the file should be automatically decoded.

```
- (BOOL)download:(NSURLOnload *)download shouldDecodeSourceDataOfMIMEType:(NSString *)encodingType
```

Parameters

download

The URL download object sending the message.

encodingType

The type of encoding used by the downloaded file. The supported encoding formats are MacBinary ("application/macbinary"), Binhex ("application/mac-binhex40") and gzip ("application/gzip").

Return Value

YES to decode the file, NO otherwise.

Special Considerations

The delegate may receive this message more than once if the file has been encoded multiple times. This method is not called if the downloaded file is not encoded.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

download:willResumeWithResponse:fromByte:

Sent when a download object has received a response from the server after attempting to resume a download.

```
- (void)download:(NSURLOnload *)download willResumeWithResponse:(NSURLOnloadResponse *)response fromByte:(long long)startingByte
```

Parameters

download

The URL download object sending the message.

response

The URL response received from the server in response to an attempt to resume a download.

startingByte

The location of the start of the resumed data, in bytes.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSURLOnload.h

download:willSendRequest:redirectResponse:

Sent when the download object determines that it must change URLs in order to continue loading a request.

```
- (NSURLOnloadRequest *)download:(NSURLOnload *)download willSendRequest:(NSURLOnloadRequest *)request redirectResponse:(NSURLOnloadResponse *)redirectResponse
```

Parameters

download

The URL download object sending the message.

request

The proposed redirected request. The delegate should inspect the redirected request to verify that it meets its needs, and create a copy with new attributes to return to the connection if necessary.

redirectResponse

The URL response that caused the redirect. May be *nil* in cases where this method is not being sent as a result of involving the delegate in redirect processing.

Return Value

The actual URL request to use in light of the redirection response. The delegate may copy and modify *request* as necessary to change its attributes, return *request* unmodified, or return *nil*.

Discussion

If the delegate wishes to cancel the redirect, it should call the *download* object's [cancel](#) (page 8) method. Alternatively, the delegate method can return *nil* to cancel the redirect, and the download will continue to process. This has special relevance in the case where *redirectResponse* is not *nil*. In this case, any data that is loaded for the download will be sent to the delegate, and the delegate will receive a [downloadDidFinish:](#) (page 18) or [download:didFailWithError:](#) (page 14) message, as appropriate.

Special Considerations

The delegate can receive this message as a result of transforming a request's URL to its canonical form, or for protocol-specific reasons, such as an HTTP redirect. The delegate implementation should be prepared to receive this message multiple times.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

downloadDidBegin:

Sent immediately after a download object begins a download.

```
- (void)downloadDidBegin:(NSURLOnload *)download
```

Parameters

download

The URL download object sending the message.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLOnload.h

downloadDidFinish:

Sent when a download object has completed downloading successfully and has written its results to disk.

```
- (void)downloadDidFinish:(NSURLOnload *)download
```

Parameters

download

The URL download object sending the message.

Discussion

The delegate will receive no further messages for *download*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLDownload.h

downloadShouldUseCredentialStorage:

Sent to determine whether the URL loader should consult the credential storage to authenticate the download.

- (BOOL)downloadShouldUseCredentialStorage:(NSURLDownload *)*download*

Parameters

connection

The connection sending the message.

Discussion

This method is called before any attempt to authenticate is made. By returning `NO`, the delegate tells the download not to consult the credential storage and makes itself responsible for providing credentials for any authentication challenges. Not implementing this method is the same as returning `YES`. The delegate is free to consult the credential storage itself when it receives a [download:didReceiveAuthenticationChallenge:](#) (page 14) message.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSURLDownload.h

Document Revision History

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

Date	Notes
2009-08-12	Added API information for credential storage and protection space authentication.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

cancel **instance method 8**
canResumeDownloadDecodedWithEncodingMIMETYPE: **class method 8**

D

deletesFileUponFailure **instance method 9**
download:canAuthenticateAgainstProtectionSpace: **<NSObject> delegate method 12**
download:decideDestinationWithSuggestedFilename: **<NSObject> delegate method 13**
download:didCancelAuthenticationChallenge: **<NSObject> delegate method 13**
download:didCreateDestination: **<NSObject> delegate method 14**
download:didFailWithError: **<NSObject> delegate method 14**
download:didReceiveAuthenticationChallenge: **<NSObject> delegate method 14**
download:didReceiveDataOfLength: **<NSObject> delegate method 15**
download:didReceiveResponse: **<NSObject> delegate method 16**
download:shouldDecodeSourceDataOfMIMETYPE: **<NSObject> delegate method 16**
download:willResumeWithResponse:fromByte: **<NSObject> delegate method 17**
download:willSendRequest:redirectResponse: **<NSObject> delegate method 17**
downloadDidBegin: **<NSObject> delegate method 18**
downloadDidFinish: **<NSObject> delegate method 18**
downloadShouldUseCredentialStorage: **<NSObject> delegate method 19**

I

initWithRequest:delegate: **instance method 9**
initWithResumeData:delegate:path: **instance method 10**

R

request **instance method 10**
resumeData **instance method 10**

S

setDeletesFileUponFailure: **instance method 11**
setDestination:allowOverwrite: **instance method 11**