Documentation Language: Swift

Network Extension / NEURLFilterManager

Class

NEURLFilterManager

A class you use to configure and control a URL filter.

iOS 26.0+ | iPadOS 26.0+ | Mac Catalyst 26.0+ | macOS 26.0+

```
class NEURLFilterManager
```

Overview

The system performs URL filtering on your behalf according to your configuration and URL data set. The system filters all URL requests initiated with the WebKit and `URLSession` APIs.

During URL filtering, the system performs sub-URL generation to enumurate all possible sub-URLs for the URL in question. For example, the URL

```
https://www.example.com/a/b/c?id=123#fragment
```

parses into the following sub-URLs:

- example.com
- example.com/
- www.example.com
- www.example.com/
- example.com/a
- example.com/a/
- www.example.com/a

- www.example.com/a/
- example.com/a/b
- example.com/a/b/
- www.example.com/a/b
- www.example.com/a/b/
- example.com/a/b/c
- example.com/a/b/c/
- www.example.com/a/b/c
- www.example.com/a/b/c/
- example.com/a/b/c?id=123
- example.com/a/b/c?id=123/
- www.example.com/a/b/c?id=123
- example.com/a/b/c?id=123#fragment
- www.example.com/a/b/c?id=123/
- www.example.com/a/b/c?id=123#fragment
- example.com/a/b/c?id=123#fragment/
- www.example.com/a/b/c?id=123#fragment/

The manager matches each of these sub-URLs against your Bloom filter and then against the PIR URL database if there’s a Bloom filter match. The verdict indicates if the app should block the requested URL. Note that the manager puny-codes the requested URL before parsing. Because of this, be sure to puny-code your own URL dataset before constructing your Bloom filter and PIR database. Neither the Bloom filter nor PIR supports wildcards or regular expressions.

Instances of this class are thread-safe.

Topics

Obtaining the shared instance

```
static var shared: NEURLFilterManager
```

The shared filter manager instance your app uses to manage URL filters.

Working with a Private Information Retrieval server

```
var pirServerURL: URL?
```

A URL that contains the domain name of the PIR server.

```
var pirPrivacyPassIssuerURL: URL?
```

A URL that contains the domain name of Privacy Pass issuer.

```
var pirAuthenticationToken: String?
```

A PIR per-user authentication token string.

```
func refreshPIRParameters() async throws
```

Refetches PIR parameters from the server.

```
func resetPIRCache() async throws
```

Resets the PIR on-device cache.

Working with the filter configuration

```
func setConfiguration(pirServerURL: URL, pirPrivacyPassIssuerURL: URL?,
pirAuthenticationToken: String, controlProviderBundleIdentifier: String
) throws
```

Sets a URL filter configuration with the required PIR server attributes and app extension information.

```
func loadFromPreferences() async throws
```

Loads the current URL filter configuration from the caller’s URL filter preferences.

```
func saveToPreferences() async throws
```

Saves the URL filter configuration in the caller’s URL filter preferences.

```
func removeFromPreferences() async throws
```

Removes the URL filter configuration from the caller’s URL filter preferences.

```
func handleConfigChange() -> any AsyncSequence<Bool, Never>
```

Sets up an observer for the filter configuration change notification and models it as an asynchronous sequence.

Working with filter status

```
var status: NEURLFilterManager.Status
```

The current status of the URL filter.

```
func handleStatusChange() -> any AsyncSequence<NEURLFilterManager.Status, Never>
```

Sets up an observer for the status notification and models it as an asynchronous sequence.

```
enum Status
```

An enumeration of URL filter status codes.

Managing filter life cycle

```
var isEnabled: Bool
```

A Boolean value that toggles the enabled status of the URL filter.

```
var shouldFailClosed: Bool
```

A Boolean value that determines how the filter behaves if it fails to make a filtering decision.

```
var prefilterFetchInterval: TimeInterval
```

The time interval at which the the filter control provider app extension runs.

Identifying app and extension bundles

```
var appBundleIdentifier: String?
```

The bundle identifier of the URL filter app.

```
var controlProviderBundleIdentifier: String?
```

The bundle identifier of the filter controller provider app extension.

Handling errors

```
var lastDisconnectError: NEURLFilterManager.Error?
```

The most recent error that caused the URL Filter to stop.

```
enum Error
```

An enumeration of URL filter error codes

Describing the filter

```
var localizedDescription: LocalizedStringResource?
```

A string containing a description of the URL filter.

Relationships

Conforms To

`Sendable`, `SendableMetatype`

See Also

URL filters

`protocol NEURLFilterControlProvider`

A protocol that defines an object that’s responsible for fetching pre-filter data.

`class NEURLFilterControlProviderConfiguration`

A class that defines app extension configurations for the URL Filter control provider app extension.

`class NEURLFilter`

A class used to voluntarily validate URLs for apps that don’t use WebKit or the URL session API.

{} Filtering traffic by URL

Perform fast and robust filtering of full URLs by managing URL filtering configurations.