Class

NWParameters

An object that stores the protocols to use for connections, options for sending data, and network path constraints.

iOS 12.0+ | iPadOS 12.0+ | Mac Catalyst 12.0+ | macOS 10.14+ | tvOS 12.0+ | visionOS 1.0+ | watchOS 5.0+

```
final class NWParameters
```

Mentioned in

Indicating the source of network activity

Topics

Creating Parameters

`class var tls: NWParameters`

A set of default parameters for connections and listeners that use TLS and TCP.

`class var tcp: NWParameters`

A set of default parameters for connections and listeners that use TCP.

`class var dtls: NWParameters`

A set of default parameters for connections and listeners that use DTLS and UDP.

`class var udp: NWParameters`

A set of default parameters for connections and listeners that use UDP.

`class func quic(alpn: [String]) -> NWParameters`

Returns a set of default parameters for connections and listeners that use QUIC, with a set of supported Application-Layer Protocol Negotiation values.

`class func quicDatagram(alpn: [String]) -> NWParameters`

Returns a set of default parameters for connections and listeners that use QUIC datagrams, with a set of supported Application-Layer Protocol Negotiation values.

`convenience init(tls: NWProtocolTLS.Options?, tcp: NWProtocolTCP.Options)`

Initializes parameters for TLS connections and listeners with custom TLS and TCP options.

`convenience init(dtls: NWProtocolTLS.Options?, udp: NWProtocolUDP.Options)`

Initializes parameters for DTLS connections and listeners with custom DTLS and UDP options.

`convenience init(quic: NWProtocolQUIC.Options)`

Initializes parameters for QUIC connections and listeners with custom QUIC options.

`init()`

Initializes parameters for connections, listeners, and browsers with no protocols specified.

`init(customIPProtocolNumber: UInt8)`

Initializes parameters for connections and listeners using a custom IP protocol.

`func copy() -> NWParameters`

Performs a deep copy of a parameters object.

Modifying Protocol Stacks

`var defaultProtocolStack: NWParameters.ProtocolStack`

The protocol stack used by connections and listeners.

`class ProtocolStack`

An ordered set of protocol options that define the protocols that connections and listeners use.

`class NWProtocol`

The abstract superclass used by Network framework protocols and by custom network protocols that you define.

Selecting Paths

`var requiredInterfaceType: NWInterface.InterfaceType`

An interface type to require on connections and listeners.

`var requiredInterface: NWInterface?`

A specific interface to require on connections, listeners, and browsers.

`var requiredLocalEndpoint: NWEndpoint?`

A specific local IP address and port to use for connections and listeners.

`var prohibitConstrainedPaths: Bool`

A Boolean that prevents connections, listeners, and browsers from using network paths marked as constrained by Low Data Mode.

`var prohibitExpensivePaths: Bool`

A Boolean that prevents connections, listeners, and browsers from using network paths marked as expensive.

`var prohibitedInterfaceTypes: [NWInterface.InterfaceType]?`

A list of interface types that connections, listeners, and browsers will not use.

`var prohibitedInterfaces: [NWInterface]?`

A list of specific interfaces that connections and listeners will not use.

Customizing Connection Options

`var multipathServiceType: NWParameters.MultipathServiceType`

An option to allow connections to use multipath protocols.

`enum MultipathServiceType`

Modes in which a connection can support multipath protocols.

`var serviceClass: NWParameters.ServiceClass`

The traffic characteristics network connections send and receive.

`enum ServiceClass`

Indicates how the system prioritizes transmitted traffic by your latency and throughput needs.

`var allowFastOpen: Bool`

A Boolean that enables sending application data with protocol handshakes.

`var expiredDNSBehavior: NWParameters.ExpiredDNSBehavior`

A behavior that defines how expired DNS answers will be used.

`enum ExpiredDNSBehavior`

Options for configuring how expired DNS answers should be used.

`var requiresDNSSECValidation: Bool`

A Boolean value that determines whether a connection requires DNSSEC validation when resolving endpoints.

`var preferNoProxies: Bool`

A Boolean that indicates that connections should ignore proxies when they are enabled on the system.

`var includePeerToPeer: Bool`

A Boolean that enables peer-to-peer link technologies for connections and listeners.

`var allowLocalEndpointReuse: Bool`

A Boolean that allows reusing local addresses and ports across connections.

`var acceptLocalOnly: Bool`

A Boolean that restricts listeners to only accepting connections from the local link.

Configuring Privacy Settings

`func setPrivacyContext(NWParameters.PrivacyContext)`

Associates a privacy context with any connections or listeners that use the parameters.

`class PrivacyContext`

An object that defines the privacy requirements for a set of connections.

Instance Properties

`var allowUltraConstrainedPaths: Bool`

Allow connection to use interfaces considered ultra-constrained by the system

`var attribution: NWParameters.Attribution`

`var wifiAware: WAParameters`

Get and set Wi-Fi Aware specific connection parameters.

Instance Methods

`func wifiAware((inout WAParameters) -> Void) -> Self`

Configure Wi-Fi Aware properties on an `NWParameters` object.

Type Properties

`class var applicationService: NWParameters`

The default parameters for connecting with other, local devices that are running your app.

Enumerations

`enum Attribution`

Relationships

Conforms To

`Copyable`
`CustomDebugStringConvertible`
`NWParametersProvider`
`Sendable`
`SendableMetatype`

See Also

Essentials

`enum NWEndpoint`

A local or remote endpoint in a network connection.