Article

Adopting the Verifier API in your iPhone app

Configure and test ID Verifier support in your app for reading mobile documents.

Overview

ProximityReader enables support for the Verifier API, which allows your app to read mobile documents such as driver's licenses, state IDs, and national IDs without any additional hardware. After configuration, `ProximityReader` can begin a `MobileDocumentReaderSession` to read a mobile document. A `MobileDocumentReaderSession` brings up a sheet that collects and returns the document information you need to perform age and identity verification.

To enable the Verifier API, verify that the person's iPhone supports it, then prepare the device to read mobile documents.

For design guidance, see Human Interface Guidelines > ID Verifier.

Important

To perform mobile document requests, add the "Verifier API" capability to your app's target in Xcode. For more information, see Adding capabilities to your app.

Verify device support

To support the Verifier API, a person's iPhone requires iOS 17 and later. The following example demonstrates how to check if the person's device supports the Verifier API:

```
actor MyActor {
```

```
    // The mobile document reader you use to prepare the device and create a reader
    private let reader: MobileDocumentReader

    // Returns `nil` if the Verifier API isn't supported on this device.
    init?() {
        guard MobileDocumentReader.isSupported else {
            // This device doesn't support the Verifier API.
            return nil
        }

        self.reader = MobileDocumentReader()
    }

}
```

If the device doesn't support the Verifier API, the framework throws a `MobileDocumentReaderError.notSupported` error.

Prepare a device to read mobile documents

Before you can perform mobile document requests, your app needs to prepare the device by instantiating a `MobileDocumentReader` object and calling its `prepare(using:)` method.

If you want to show your brand's name and logo in the system UI during a document request, pass a reader token generated from your server into the prepare method. For more information about reader tokens and how to generate on your server, see Generating reader tokens for the Verifier API.

If the call to `prepare(using:)` is successful, the framework returns a `MobileDocumentReaderSession` object that you can use to read mobile documents. If the call fails, you — or the person using your app — may need to take action, depending on the reasons for the failure. Review `MobileDocumentReaderError` for more details of the kinds of errors that you can encounter.

The following example shows how to prepare device to read mobile documents:

```
actor MyActor {

    // The mobile document reader, you use to prepare the device and retrieve a read
    private let reader: MobileDocumentReader

    // The current reader session. If nil, prepare the device.
    private var session: MobileDocumentReaderSession?
```

```
    func prepareDevice() async {
        // Check whether the device is already prepared to read mobile documents.
        guard self.session == nil else {
            return
        }

        // Fetch the reader configuration. This requires network access.
        let configuration = try await self.reader.configuration

        // Get the current reader instance identifier -- do not cache this value dir
        let identifier = configuration.readerInstanceIdentifier

        // Send the identifier to your server in exchange for a reader token.
        let tokenString = try await MyWebService().fetchToken(for: identifier)

        // Create a reader token from the token string.
        let readerToken = MobileDocumentReader.Token(tokenString)

        do {
            self.session = try await self.reader.prepare(using: readerToken)
        } catch MobileDocumentReader.invalidToken {
            print("The reader token provided to prepare was invalid.")
        } catch {
            print("An error occurred while preparing the reader: \(error.localizedDe
        }
    }
```

The first time you use a reader token to prepare a session, the framework performs a network call to the Verifier API server. The framework can perform subsequent prepare calls with the same reader token offline. After you use a reader token to successfully prepare the device and create a `MobileDocumentReaderSession`, you can reuse it to prepare the device repeatedly for up to 48 hours.

> **Note**
>
> If you reuse a reader token after the 48-hour period has elapsed, the framework throws a `MobileDocumentReaderError.sessionExpired` error.

Read mobile documents

Use `MobileDocumentReaderSession` to present a system UI to prompt the person using your app to present their mobile document to the reader and approve the request. After successfully reading the mobile document, the system UI either shows the document information onscreen for review or dismisses it entirely, returning the response to your app. Whether the system UI displays the document information onscreen, or returns the data to your app depends on which `MobileDocumentRequest` your app performs.

If a read isn't successful, the framework dismisses the system UI and `requestDocument(_:)` throws a `MobileDocumentReaderError`. If the person dismissed the system UI, the framework throws `MobileDocumentReaderError.cancelled`.

Stop reading mobile documents

When you're finished reading mobile documents, you can invalidate the reader session by deallocating it.

```
actor MyActor {

    // A currently valid, prepared session.
    private var session: MobileDocumentReaderSession?

    // Invalidate the reader session by deallocating it.
    func stopReadingDocuments() {
        self.session = nil
    }

}
```

Test your implementation

Even if you don't live in an area that supports mobile documents, you can test your implementation in the Simulator or by downloading a developer profile on your local device.

When you perform a document request, the system simulates a successful document read and displays or returns a mock response.

To enable the simulated test mode on your iPhone, download and install the "Mobile Document Reader (Developer)" profile from Profiles and Logs.

See Also

Mobile document reader

Generating reader tokens for the Verifier API

Configure your server to generate reader tokens to prepare a device for mobile document reading.

{} Checking IDs with the Verifier API

Read and verify mobile driver's license information without any additional hardware.

`class MobileDocumentReader`

An object for configuring mobile document reading on the current device.

`class MobileDocumentReaderSession`

The object you use to start reading a mobile document.