What’s New in Authentication

Easier and more secure sign-in

Ricky Mondello, iOS Engineer
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Use your Apple ID "jappleseed@icloud.com" to sign in to Bird and create your account with the information below.

<table>
<thead>
<tr>
<th>NAME</th>
<th>Jane Appleseed</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMAIL</td>
<td>Share My Email</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:janeappleseed@icloud.com">janeappleseed@icloud.com</a></td>
</tr>
</tbody>
</table>

Hide My Email
Forward To: janeappleseed@icloud.com

Continue
Easier than passwords
Sign In with Apple accounts are secure, verified accounts
Use your saved password to sign in to Shiny?

Saved from shiny.example.com  rmondello@example.com

Continue
Introducing Sign In with Apple

Sign in with Apple

WWDC 2019
Sign In with Apple

Password-based authentication

Warnings for weak passwords

OAuth sign-in

USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
// https://example.com/.well-known/apple-app-site-association

{
   "webcredentials": {
   }
}

// https://example.com/.well-known/apple-app-site-association

{
    "webcredentials": {
    }
}
// https://example.com/.well-known/apple-app-site-association

{
  "webcredentials": {
    "apps": [
    ]
  }
}
// https://example.com/.well-known/apple-app-site-association
// Universal Links

{
    "applinks": {
        "apps": [],
        "details": [
            {
                "components": [ { "/": "/a/path/*" } ]
            }
        ]
    }
}
Universal Links

Handling multiple apps (iOS 13, macOS 10.15)

```json
{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                            "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
                "components": [ { "": "/a/path/*" } ]
            }
        ]
    }
}
```
// https://example.com/.well-known/apple-app-site-association
// Universal Links
// Handling multiple apps (iOS 13, macOS 10.15)

{
  "applinks": {
    "apps": [],
    "details": [
      {
        "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                     "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
        "components": [ { "/": "/a/path/*" } ]
      }
    ]
  }
}
// https://example.com/.well-known/apple-app-site-association
// Universal Links
// Maintaining compatibility with iOS prior to iOS 13

{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                            "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
                "components": [ { "/": "/a/path/*" } ]
            }
        ]
    }
}
// https://example.com/.well-known/apple-app-site-association
// Universal Links
// Maintaining compatibility with iOS prior to iOS 13

{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                            "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
                "components": [ { "/": "/a/path/*" } ]
            }
        ]
    }
}
// https://example.com/.well-known/apple-app-site-association
// Universal Links
// Maintaining compatibility with iOS prior to iOS 13

{
  "applinks": {
    "apps": [],
    "details": [
      {
        "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                    "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
        "components": [ { "/": "/a/path/*" } ]
      }
    ]
  }
}
// https://example.com/.well-known/apple-app-site-association
// Universal Links
// Maintaining compatibility with iOS prior to iOS 13

{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appIDs": [ "A1B2C3D4E5.com.example.Shiny",
                           "A1B2C3D4E5.uikitformac.com.example.Shiny" ],
                "components": [ { "/": "/a/path/*" } ]
            }
        ]
    }
}

What's New in Universal Links

WWDC 2019
// Requesting a password-based account for sign-in

import AuthenticationServices
// Requesting a password-based account for sign-in

import AuthenticationServices
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationPasswordProvider().createRequest()
])
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationPasswordProvider().createRequest()
])
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [ASAuthorizationPasswordProvider().createRequest()])

controller.delegate = self
controller.presentationContextProvider = self
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationPasswordProvider().createRequest()
])

controller.delegate = self
controller.presentationContextProvider = self
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationPasswordProvider().createRequest()
])

controller.delegate = self
controller.presentationContextProvider = self

controller.performRequests()
// Requesting a password-based account for sign-in

import AuthenticationServices

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationPasswordProvider().createRequest()
])

controller.delegate = self
controller.presentationContextProvider = self

controller.performRequests()
// Requesting a password-based account for sign-in

// ASAuthorizationControllerDelegate
func authorizationController(controller: ASAuthorizationController,
    didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASPasswordCredential {
        // Use credential.user and credential.password to sign in.
    }
}

// Requesting a password-based account for sign-in

// ASAuthorizationControllerDelegate
func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASPasswordCredential {
        // Use credential.user and credential.password to sign in.
    }
}
// Requesting a password-based account for sign-in

// ASAuthorizationControllerDelegate

func authorizationController(controller: ASAuthorizationController,
    didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASPasswordCredential {
        // Use credential.user and credential.password to sign in.
    }
}

func authorizationController(controller: ASAuthorizationController,
    didCompleteWithError error: Error) {
    // Fall back to login UI.
}
// Requesting a password-based account for sign-in

// ASAuthorizationControllerDelegate

func authorizationController(controller: ASAuthorizationController,
    didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASPasswordCredential {
        // Use credential.user and credential.password to sign in.
    }
}

func authorizationController(controller: ASAuthorizationController,
    didCompleteWithError error: Error) {
    // Fall back to login UI.
}
Associated Domains

Introducing Password AutoFill for Apps

What’s New in Universal Links

WWDC 2017

WWDC 2019
// Requesting both Sign In with Apple and password-based accounts for sign-in

let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationAppleIDProvider().createRequest(),
    ASAuthorizationPasswordProvider().createRequest()
])

controller.delegate = self
controller.presentationContextProvider = self

controller.performRequests()
let controller = ASAuthorizationController(authorizationRequests: [
    ASAuthorizationAppleIDProvider().createRequest(),
    ASAuthorizationPasswordProvider().createRequest()
])

controller.delegate = self
controller.presentationContextProvider = self

controller.performRequests()
// Requesting both Sign In with Apple and password-based accounts for sign-in

// ASAuthorizationControllerDelegate
func authorizationController(controller: ASAuthorizationController,
                              didCompleteWithAuthorization authorization: ASAuthorization) {
    switch authorization.credential {
        case let credential as ASAuthorizationAppleIDCredential:
            // Sign the user in using the Apple credential.

        case let credential as ASPasswordCredential:
            // Sign the user in using the password credential.

        default:
            // Fall back to login UI.
    }
}
// Requesting both Sign In with Apple and password-based accounts for sign-in

// ASAuthorizationControllerDelegate
func authorizationController(controller: ASAuthorizationController,
   didCompleteWithAuthorization authorization: ASAuthorization) {

switch authorization.credential {
    case let credential as ASAuthorizationAppleIDCredential:
        // Sign the user in using the Apple credential.

    case let credential as ASPasswordCredential:
        // Sign the user in using the password credential.

    default:
        // Fall back to login UI.
}
}
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
create account

Email
reza@example.com

Password
remSa3-takpe-

Sign Up

iPhone created a strong password for this app.
This password will be saved to your iCloud Keychain and will
AutoFill on all your devices. You can look up your saved
passwords in Settings or by asking Siri.

Use Strong Password

Choose My Own Password
Welcome to Twitter!

This is the best place to see what's happening in your world. Find some people and topics to follow now.

Let's go!

This password may be easily guessed and should be changed to a strong password.

iPhone can create a strong password for you. Would you like to change your password for "twitter.com"?

Change Password on Website

Not Now
Change password

Current password

Forgot password?

New password

Confirm password

Save
https://example.com/.well-known/change-password
Well-Known URL for Changing Passwords

wicg.github.io/change-password-url
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
“Shiny” Wants to Use “example.com” to Sign In

This allows the app and website to share information about you.

Cancel  Continue

Log in with example.com

Forgot Password?
Create Account
Example Identity Provider

The Shiny app by would like to use your account.

This will give Shiny your name.

Allow Shiny access?

- [ ] Allow
  - [x] Do Not Allow

Signed in as: james@example.com
today's shiny
Example Identity Provider

The Shiny app by would like to use your account.

This will give Shiny your name.

Allow Shiny access?

Signed in as: garrett@example.com

Allow

Do Not Allow
OAuth with ASWebAuthenticationSession

More private sign-in

Supporting multiple windows

Migrating from SFAuthenticationSession
OAuth with ASWebAuthenticationSession

More private sign-in

Supporting multiple windows

Migrating from SFAuthenticationSession
Example Identity Provider

Sign in to allow the app Shiny to connect to your account.

Email

Password

Sign in

Use “rmondello@example.com”
import AuthenticationServices

func signIn() {
    let session = ASWebAuthenticationSession(url: URL(string: "https://example.com/login")!,
        callbackURLScheme: "custom-oauth-scheme",
        completionHandler: { (callbackURL, error) in
            // Handle login
    })
    session.prefersEphemeralWebBrowserSession = true
    session.start()
}

// Giving users additional privacy using ASWebAuthenticationSession
// Giving users additional privacy using ASWebAuthenticationSession

import AuthenticationServices

func signIn() {
    let session = ASWebAuthenticationSession(url: URL(string: "https://example.com/login")!,
        callbackURLScheme: "custom-oauth-scheme",
        completionHandler: { (callbackURL, error) in
            // Handle login
        })
    session.prefersEphemeralWebBrowserSession = true
    session.start()
}

NEW
OAuth with ASWebAuthenticationSession

More private sign-in

Supporting multiple windows

Migrating from SFAuthenticationSession
// Providing a window to the authentication session

import AuthenticationServices

func signIn() {
    let session = ASWebAuthenticationSession(url: URL(string: "https://example.com/login"),
        callbackURLScheme: "custom-oauth-scheme",
        completionHandler: { (callbackURL, error) in
            // Handle login
        })
    session.presentationContextProvider = self
    session.start()
}

func presentationAnchor(for _: ASWebAuthenticationSession) -> ASPresentationAnchor {
    return /* the relevant window */
}
/ Providing a window to the authentication session

import AuthenticationServices

func signIn() {
    let session = ASWebAuthenticationSession(url: URL(string: "https://example.com/login")!,
        callbackURLScheme: "custom-oauth-scheme",
        completionHandler: { (callbackURL, error) in
            // Handle login
        })
        session.presentationContextProvider = self
    session.start()
}

func presentationAnchor(for _: ASWebAuthenticationSession) -> ASPresentationAnchor {
    return /* the relevant window */
}

// Providing a window to the authentication session

import AuthenticationServices

func signIn() {
    let session = ASWebAuthenticationSession(url: URL(string: "https://example.com/login")!,
        callbackURLScheme: "custom-oauth-scheme",
        completionHandler: { (callbackURL, error) in
            // Handle login
        })
    session.presentationContextProvider = self
    session.start()
}

func presentationAnchor(for _: ASWebAuthenticationSession) -> ASPresentationAnchor {
    return /* the relevant window */
}
OAuth with ASWebAuthenticationSession

More private sign-in

Supporting multiple windows

Migrating from SFAuthenticationSession
ASWebAuthenticationSession
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
FIDO2-compliant Security Key
FIDO2-compliant Security Key
Security Key
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Sign In with Apple
Password-based authentication
Warnings for weak passwords
OAuth sign-in
USB security keys on macOS
Summary

Adopt Sign In with Apple
Summary

Adopt Sign In with Apple

Make password sign-in easy
Summary

Adopt Sign In with Apple

Make password sign-in easy

Implement /.well-known/change-password
More Information

developer.apple.com/wwdc19/516