Implementing AutoFill Credential Provider Extensions

Session 721

David Quesada, iOS Engineer
Password AutoFill

AutoFill Integration for Password Manager Apps

Best Practices
Password AutoFill
AutoFill Integration for Password Manager Apps
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Configure Your Project
Enable the ‘AutoFill Credential Provider’ capability

Steps: ✓ Add the AutoFill Credential Provider feature to your App ID.
✓ Add the AutoFill Credential Provider entitlement to your entitlements file
✓ Link AuthenticationServices.framework
Configure Your Project
Add an AutoFill Credential Provider extension
import AuthenticationServices

class CredentialProviderViewController: ASCredentialProviderViewController {
}

Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Implement a Credential List

Overview

AutoFill
Implement a Credential List

Overview

AutoFill

Your App Extension
Implement a Credential List

Overview

class ASCredentialServiceIdentifier: NSObject {

    enum IdentifierType {
        case domain
        case URL
    }

    var identifier: String { get }
    var type: ASCredentialServiceIdentifierType { get }
}
Implement a Credential List

Overview

AUTOFill

ASCredentialServiceIdentifier

.type = .domain
.identifier = "shiny.example.com"

Your App Extension
Implement a Credential List

Overview

```swift
class ASCredentialProviderViewController: UIViewController {
    func prepareCredentialList(for serviceIdentifiers: [ASCredentialServiceIdentifier]) {
        // Show the credential list UI
    }
}
```
Implement a Credential List

Overview

```swift
self.extensionContext.cancelRequest(withError:
NSError(domain: ASExtensionErrorDomain,
    code: ASExtensionError.userCanceled.rawValue))
```
Overview

```swift
self.extensionContext.cancelRequest(withError:
NSError(domain: ASExtensionErrorDomain,
code: ASExtensionError.userCanceled.rawValue))
```
Implement a Credential List

Overview

```swift
self.extensionContext.cancelRequest(withError:
    NSError(domain: ASExtensionErrorDomain,
            code: ASExtensionError.userCanceled.rawValue))
```
Implement a Credential List

Overview

class ASPasswordCredential: NSObject {
    var user: String { get }
    var password: String { get }
}

let selectedCredential = /* Create an ASPasswordCredential based on the user's selection. */

self.extensionContext.completeRequest(
    withSelectedCredential: selectedCredential,
    completionHandler: nil)
Implement a Credential List

Overview

```swift
class ASPasswordCredential: NSObject {
  var user: String { get }
  var password: String { get }
}

let selectedCredential = /* Create an ASPasswordCredential based on the user's selection. */

self.extensionContext.completeRequest(
  withSelectedCredential: selectedCredential,
  completionHandler: nil)
```
Implement a Credential List

Overview

AutoFill

"j_appleseed"
"tuxtad-mowma6-Mifneb"
Implement a Credential List

Best practices
Implement a Credential List

Best practices

Add a cancel button
Implement a Credential List

Best practices

Add a cancel button

Allow users to see all their credentials
Implement a Credential List

Best practices

Add a cancel button

Allow users to see all their credentials

Authenticate the user if needed
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Show Credentials in the QuickType Bar

Overview
Show Credentials in the QuickType Bar

Overview

Your App

j_appleseed@example.com

AutoFill
Show Credentials in the QuickType Bar

Overview

Your App

AutoFill

j_appleseed@example.com
Show Credentials in the QuickType Bar

Overview

Your App

AutoFill

j_appleseed@example.com

Your Extension
Show Credentials in the QuickType Bar

Overview

Your App

AutoFill

j_appleseed
example.com

Your Extension
Show Credentials in the QuickType Bar

Overview

Your App

Password Database

j_appleseed@example.com

Your Extension

AutoFill
Show Credentials in the QuickType Bar

Overview

Your App → Password Database

Password Database → j_appleseed "tuxtad-mowma6-Mifneb" → AutoFill

Your Extension
Show Credentials in the QuickType Bar

Overview

Your App

Password Database

Your Extension

AutoFill

j_appleseed
"tuxtad-mowma6-Mifneb"
Show Credentials in the QuickType Bar

Overview
Show Credentials in the QuickType Bar

What you need to do

Provide credential identities to AutoFill

Return passwords from extension

Show custom authentication UI (optional)
Show Credentials in the QuickType Bar
Provide credential identities to AutoFill

class ASPasswordCredentialIdentity: NSObject, NSCopying, NSSecureCoding {
    var serviceIdentifier: ASCredentialServiceIdentifier { get }
    var user: String { get }
    var recordIdentifier: String? { get }
    var rank: Int
    init(serviceIdentifier: ASCredentialServiceIdentifier, user: String,
         recordIdentifier: String?)
}
Show Credentials in the QuickType Bar
Provide credential identities to AutoFill

class ASPasswordCredentialIdentity: NSObject, NSCopying, NSSecureCoding {
    var serviceIdentifier: ASCredentialServiceIdentifier { get }
    var user: String { get }
    var recordIdentifier: String? { get }
    var rank: Int
    init(serviceIdentifier: ASCredentialServiceIdentifier, user: String,
         recordIdentifier: String?)
}
Show Credentials in the QuickType Bar
Provide credential identities to AutoFill

class ASPasswordCredentialIdentity: NSObject, NSCopying, NSSecureCoding {
    var serviceIdentifier: ASCredentialServiceIdentifier { get }
    var user: String { get }
    var recordIdentifier: String? { get }
    var rank: Int
    init(serviceIdentifier: ASCredentialServiceIdentifier, user: String, recordIdentifier: String?)
}
Show Credentials in the QuickType Bar
Provide credential identities to AutoFill

class ASPasswordCredentialIdentity: NSObject, NSCopying, NSSecureCoding {
    var serviceIdentifier: ASCredentialServiceIdentifier { get }
    var user: String { get }
    var recordIdentifier: String? { get }
    var rank: Int

    init(serviceIdentifier: ASCredentialServiceIdentifier, user: String,
         recordIdentifier: String?)
}
Show Credentials in the QuickType Bar
Provide credential identities to AutoFill

class ASPasswordCredentialIdentity: NSObject, NSCopying, NSSecureCoding {
    var serviceIdentifier: ASCredentialServiceIdentifier { get }
    var user: String { get }
    var recordIdentifier: String? { get }
    var rank: Int

    init(serviceIdentifier: ASCredentialServiceIdentifier, user: String,
         recordIdentifier: String?)
}
Show Credentials in the QuickType Bar
Use the credential identity store

Secure, never leaves device

Writable only by your app and extensions

Read only by AutoFill

Modifiable only when extension is enabled
Show Credentials in the QuickType Bar
Use the credential identity store

Secure, never leaves device

Writable only by your app and extensions

Read only by AutoFill

Modifiable only when extension is enabled
Show Credentials in the QuickType Bar
Use the credential identity store

Secure, never leaves device

Writable only by your app and extensions

Read only by AutoFill

Modifiable only when extension is enabled
Show Credentials in the QuickType Bar

Use the credential identity store

Secure, never leaves device

Writable only by your app and extensions

Read only by AutoFill

Modifiable only when extension is enabled
Show Credentials in the QuickType Bar
Use the credential identity store

Secure, never leaves device

Writable only by your app and extensions

Read only by AutoFill

Modifiable only when extension is enabled
Show Credentials in the QuickType Bar
Use the credential identity store

Update the store as needed
Show Credentials in the QuickType Bar

Use the credential identity store

Update the store as needed

• New credentials are available
Show Credentials in the QuickType Bar

Use the credential identity store

Update the store as needed

• New credentials are available
• Credentials are modified
Show Credentials in the QuickType Bar
Use the credential identity store

Update the store as needed
• New credentials are available
• Credentials are modified
• Credentials are no longer available
class ASCredentialIdentityStore: NSObject {

  class var shared: ASCredentialIdentityStore { get }

  func replaceCredentialIdentities(with newCredentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

  func removeAllCredentialIdentities(_ completion: ((Bool, Error?) -> Void)?)

  func saveCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity],
                                completion: ((Bool, Error?) -> Void)?)

  func removeCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity],
                                 completion: ((Bool, Error?) -> Void)?)

}
class ASCredentialIdentityStore: NSObject {

    class var shared: ASCredentialIdentityStore { get }

    func replaceCredentialIdentities(with newCredentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

    func removeAllCredentialIdentities(_ completion: ((Bool, Error?) -> Void)?)

    func saveCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

    func removeCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

}
class ASCredentialIdentityStore: NSObject {

   class var shared: ASCredentialIdentityStore { get }

   func replaceCredentialIdentities(with newCredentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

   func removeAllCredentialIdentities(_ completion: ((Bool, Error?) -> Void)?)

   func saveCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

   func removeCredentialIdentities(_ credentialIdentities: [ASPasswordCredentialIdentity], completion: ((Bool, Error?) -> Void)?)

}


Show Credentials in the QuickType Bar

Use the credential identity store

The store may be unexpectedly deleted
Show Credentials in the QuickType Bar

Use the credential identity store

The store may be unexpectedly deleted
• Extension is disabled, re-enabled
Show Credentials in the QuickType Bar
Use the credential identity store

The store may be unexpectedly deleted
• Extension is disabled, re-enabled
• App provides stale credential identities
Show Credentials in the QuickType Bar
Use the credential identity store

The store may be unexpectedly deleted
• Extension is disabled, re-enabled
• App provides stale credential identities
• User restores from backup
Show Credentials in the QuickType Bar
Use the credential identity store

The store may be unexpectedly deleted
• Extension is disabled, re-enabled
• App provides stale credential identities
• User restores from backup

ASCredentialIdentityStore can help
class ASCredentialIdentityStore: NSObject {

    func getState(_ completion: @escaping (ASCredentialIdentityStoreState) -> Void) {
    }

}

class ASCredentialIdentityStoreState: NSObject {

    var isEnabled: Bool { get }

    var supportsIncrementalUpdates: Bool { get }

}
class ASCredentialIdentityStore: NSObject {

    func getState(_ completion: @escaping (ASCredentialIdentityStoreState) -> Void)
}

class ASCredentialIdentityStoreState: NSObject {

    var isEnabled: Bool { get }

    var supportsIncrementalUpdates: Bool { get }
}

class ASCredentialIdentityStore: NSObject {

    func getState(_ completion: @escaping (ASCredentialIdentityStoreState) -> Void) {
    }
}

class ASCredentialIdentityStoreState: NSObject {

    var isEnabled: Bool { get }

    var supportsIncrementalUpdates: Bool { get }

}
class ASCredentialIdentityStore: NSObject {

    func getState(_ completion: @escaping (ASCredentialIdentityStoreState) -> Void)
}

class ASCredentialIdentityStoreState: NSObject {

    var isEnabled: Bool { get }

    var supportsIncrementalUpdates: Bool { get }

}
class CredentialProviderViewController: ASCredentialProviderViewController {

    override func provideCredentialWithoutUserInteraction(for credentialIdentity: ASPasswordCredentialIdentity) {

        // Look up the password for the selected credential identity
        let credential: ASPasswordCredential = /* ... */
        self.extensionContext.completeRequest(withSelectedCredential: credential,
                                              completionHandler: nil)
    }
}
class CredentialProviderViewController: ASCredentialProviderViewController {

    override func provideCredentialWithoutUserInteraction(for credentialIdentity: ASPasswordCredentialIdentity) {

        // Tell the system to display your UI
        self.extensionContext.cancelRequest(withError: NSError(domain: ASExtensionErrorDomain, code: ASExtensionError.userInteractionRequired.rawValue))
    }
}
class CredentialProviderViewController: ASCredentialProviderViewController {

    override func provideCredentialWithoutUserInteraction(for credentialIdentity: ASPasswordCredentialIdentity) {

        // Tell the system to display your UI
        self.extensionContext.cancelRequest(withError:
            NSError(domain: ASExtensionErrorDomain,
                code: ASExtensionError.userInteractionRequired.rawValue))
    }

    override func prepareInterfaceToProvideCredential(for credentialIdentity: ASPasswordCredentialIdentity) {

        // Set up UI for allowing the user to fill the provided credential
    }
}
class CredentialProviderViewController: ASCredentialProviderViewController {

    private func userDidAuthenticateForYourExtension() {

        let credential: ASPasswordCredential = /* ... */
        self.extensionContext.completeRequest(withSelectedCredential: credential,
                                              completionHandler: nil)

    }

}
Show Credentials in the QuickType Bar
Provide passwords from your extension

Respond to the initial request quickly
Show Credentials in the QuickType Bar
Provide passwords from your extension

Respond to the initial request quickly

Users don’t see your UI yet
Show Credentials in the QuickType Bar
Provide passwords from your extension

Respond to the initial request quickly

Users don’t see your UI yet

AutoFill will cancel the request after a timeout
Show Credentials in the QuickType Bar
Provide passwords from your extension

Respond to the initial request quickly

Users don’t see your UI yet

AutoFill will cancel the request after a timeout

• Except simulator and debug builds
Show Credentials in the QuickType Bar

Best practices

Keep the credential identity store up-to-date
Show Credentials in the QuickType Bar

Best practices

Keep the credential identity store up-to-date

Use ASCredentialIdentityStore’s incremental update APIs
Show Credentials in the QuickType Bar

Best practices

Keep the credential identity store up-to-date

Use ASCredentialIdentityStore’s incremental update APIs

Keep extension UI to a minimum
Show Credentials in the QuickType Bar

Best practices

- Keep the credential identity store up-to-date
- Use ASCredentialIdentityStore’s incremental update APIs
- Keep extension UI to a minimum
- Avoid expensive setup in every `viewDidLoad()`
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Password AutoFill Integration

Configure your project

Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings
Allow Users to Configure Your Extension in Settings

Your extension may require additional setup after being enabled
• Provide credential identities for QuickType
• Sign user into online service
• Get user authentication or confirmation for QuickType
Allow Users to Configure Your Extension in Settings

Your extension may require additional setup after being enabled

• Provide credential identities for QuickType
• Sign user into online service
• Get user authentication or confirmation for QuickType

AuthenticationServices provides API for this
Allow Users to Configure Your Extension in Settings

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Property List</td>
<td>Dictionary</td>
<td>$(DEVELOPMENT_LANGUAGE)</td>
</tr>
<tr>
<td>Localization native development region</td>
<td>String</td>
<td>$(DEVELOPMENT_LANGUAGE)</td>
</tr>
<tr>
<td>Bundle display name</td>
<td>String</td>
<td>CredentialProviderExtension</td>
</tr>
<tr>
<td>Executable file</td>
<td>String</td>
<td>$(EXECUTABLE_NAME)</td>
</tr>
<tr>
<td>Bundle identifier</td>
<td>String</td>
<td>$(PRODUCT_BUNDLE_IDENTIFIER)</td>
</tr>
<tr>
<td>InfoDictionary version</td>
<td>String</td>
<td>6.0</td>
</tr>
<tr>
<td>Bundle name</td>
<td>String</td>
<td>$(PRODUCT_NAME)</td>
</tr>
<tr>
<td>Bundle OS Type code</td>
<td>String</td>
<td>XPC!</td>
</tr>
<tr>
<td>Bundle versions string, short</td>
<td>String</td>
<td>1.0</td>
</tr>
<tr>
<td>Bundle version</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>NSExtension</td>
<td>Dictionary</td>
<td>(3 items)</td>
</tr>
<tr>
<td>NSExtensionAttributes</td>
<td>Dictionary</td>
<td>(1 item)</td>
</tr>
<tr>
<td>ASCredentialProviderExtensionShowsConfigurationUI</td>
<td>Boolean</td>
<td>YES</td>
</tr>
<tr>
<td>NSExtensionMainStoryboard</td>
<td>String</td>
<td>MainInterface</td>
</tr>
<tr>
<td>NSExtensionPointIdentifier</td>
<td>String</td>
<td>com.apple.authentication-services-credential-provider-ui</td>
</tr>
</tbody>
</table>
class CredentialProviderViewController: ASCredentialProviderViewController {

    func prepareInterfaceForExtensionConfiguration() {

        // Present appropriate Settings UI

    }

}
class CredentialProviderViewController: ASCredentialProviderViewController {

    func prepareInterfaceForExtensionConfiguration() {
        // Present appropriate Settings UI
    }

    private func myWorkIsDone() {
        // Call this to dismiss the extension UI when the user is done
        self.extensionContext.completeExtensionConfigurationRequest()
    }

}
Best Practices
Extension Development

Best practices

class ASCredentialProviderViewController {
    func prepareCredentialList(for:)
    func prepareInterfaceToProvideCredential(for:)
    func prepareInterfaceForExtensionConfiguration()
}

Use separate view controllers

Present or use view controller containment
Extension Development

Best practices

Extensions should be lightweight

Use App Groups, shared keychains
Extension Development
Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials
Extension Development
Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials
• Select your extension’s scheme
Extension Development
Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials
• Select your extension’s scheme
• Run with Safari as a target
Extension Development
Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials
• Select your extension’s scheme
• Run with Safari as a target
• Invoke your extension
Extension Development
Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials
• Select your extension’s scheme
• Run with Safari as a target
• Invoke your extension

Use “Debug > Attach to Process” for other cases
Summary

Password manager apps can integrate with Password AutoFill

Show the user’s list of credentials

Suggest the user’s credentials in the QuickType bar

Configure your app extension within Settings
More Information

https://developer.apple.com/wwdc18/721

Automatic Strong Passwords and Security Code AutoFill

Introducing Password AutoFill for Apps
Apple WWDC18