#WWDC18

# Implementing AutoFill Credential Provider Extensions

Session 721

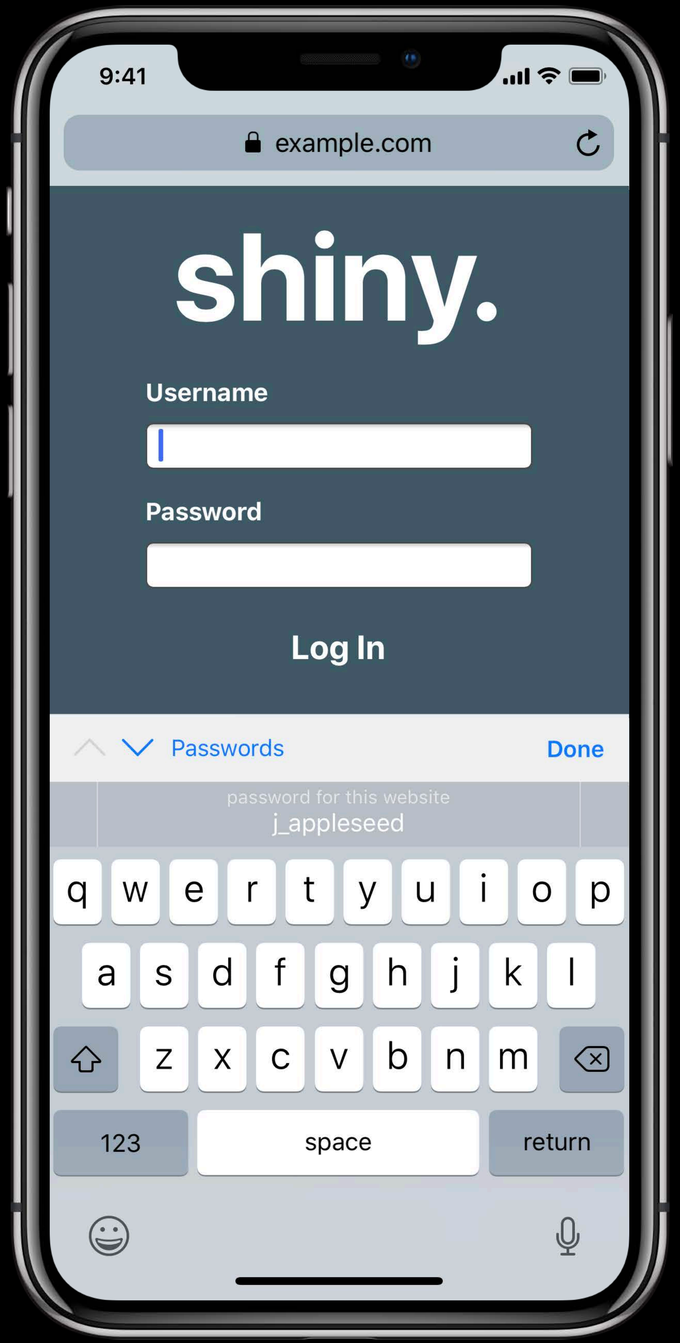
David Quesada, iOS Engineer

Password AutoFill

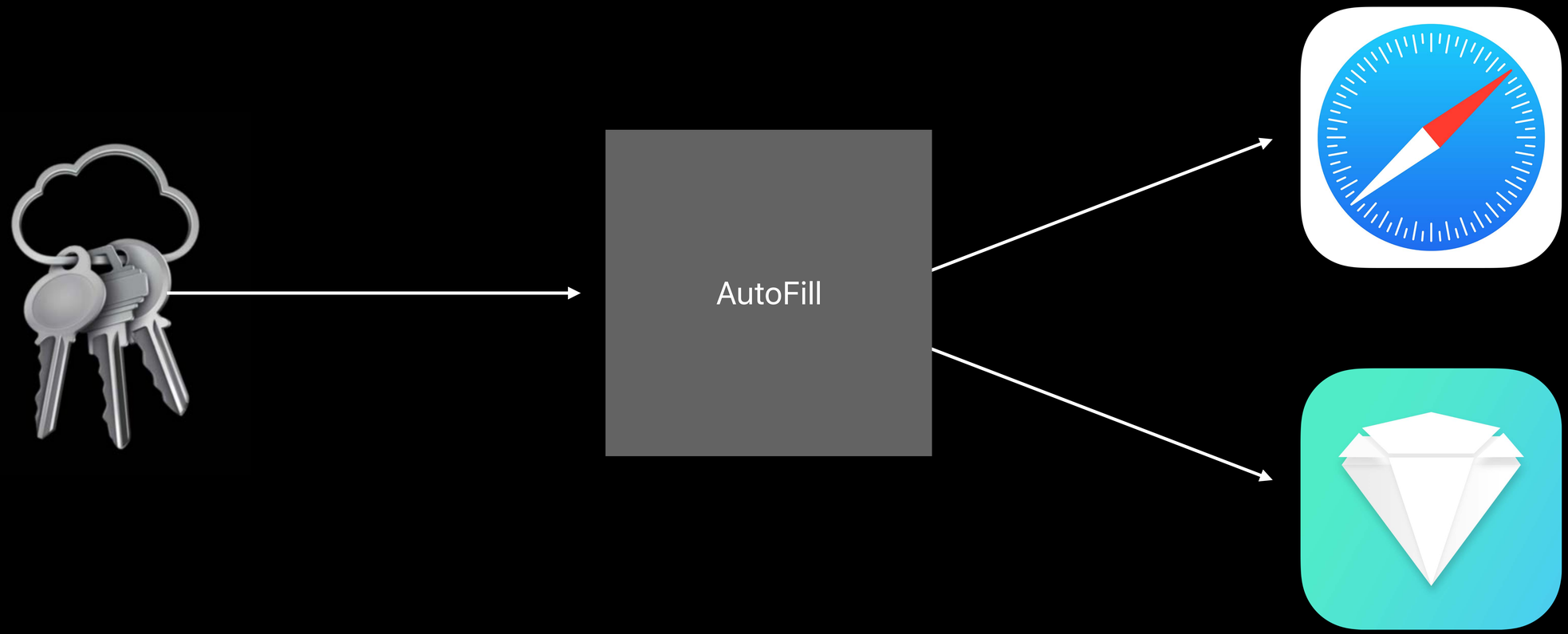
AutoFill Integration for Password Manager Apps

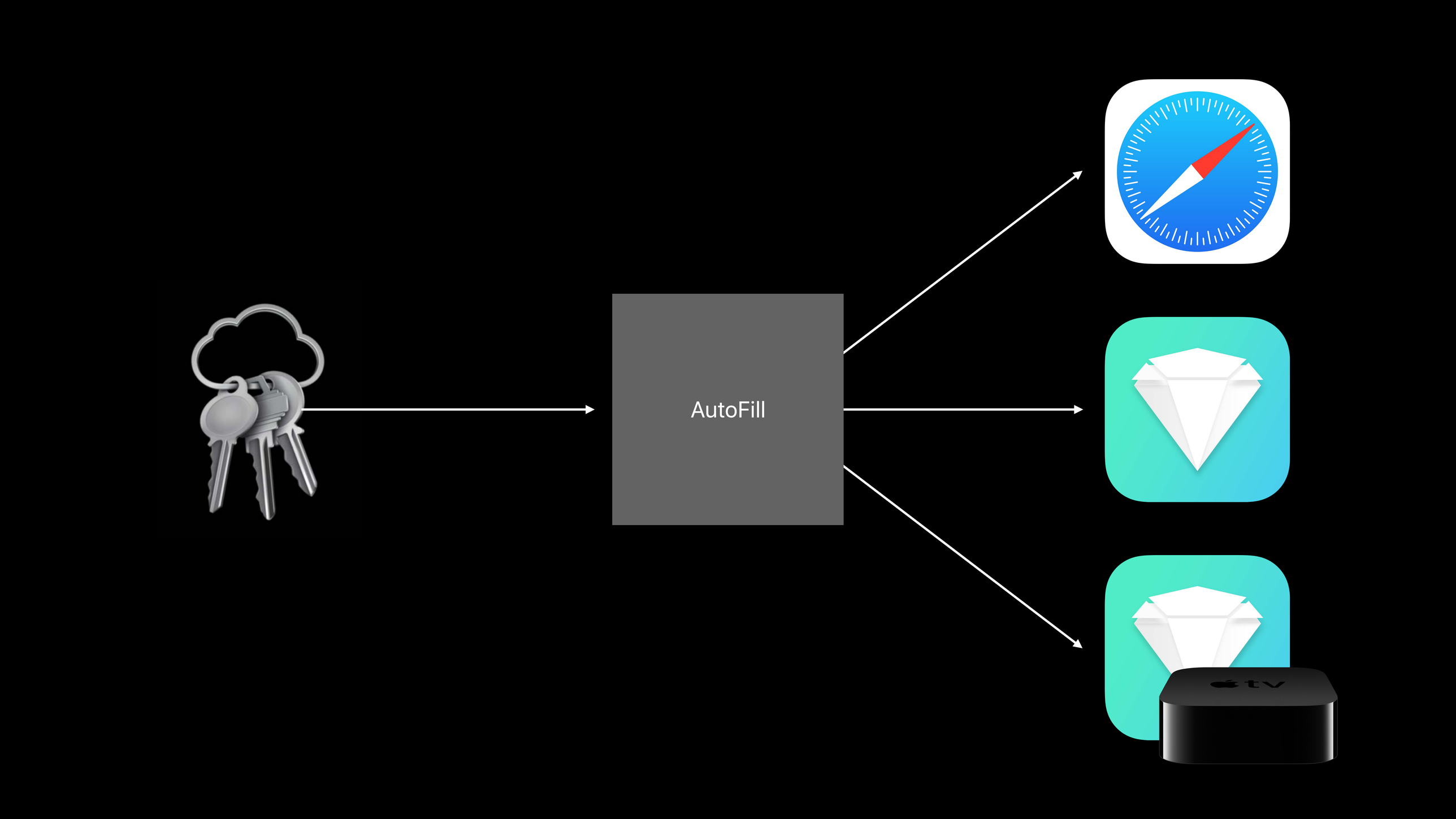
Best Practices

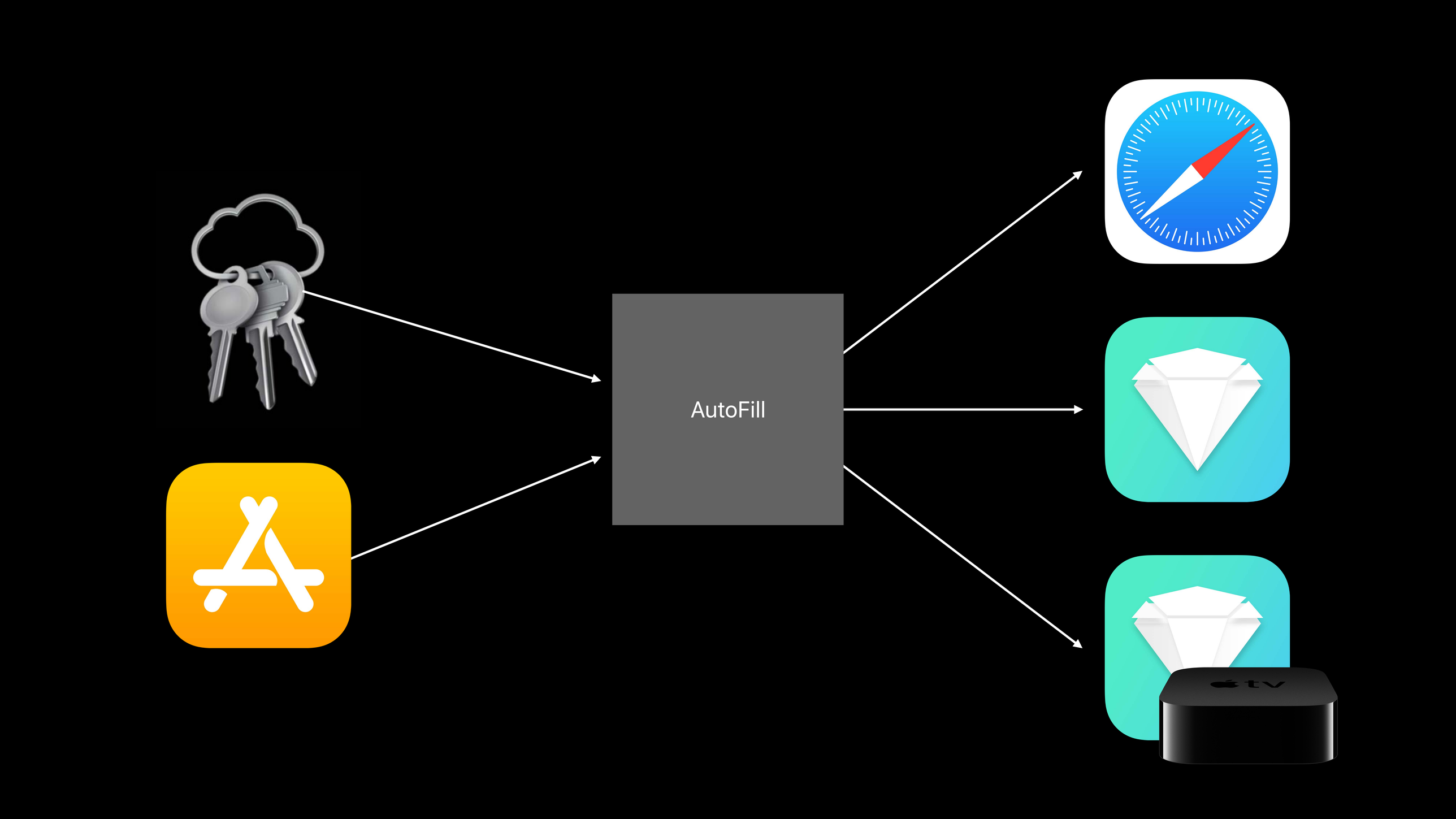
# Password AutoFill

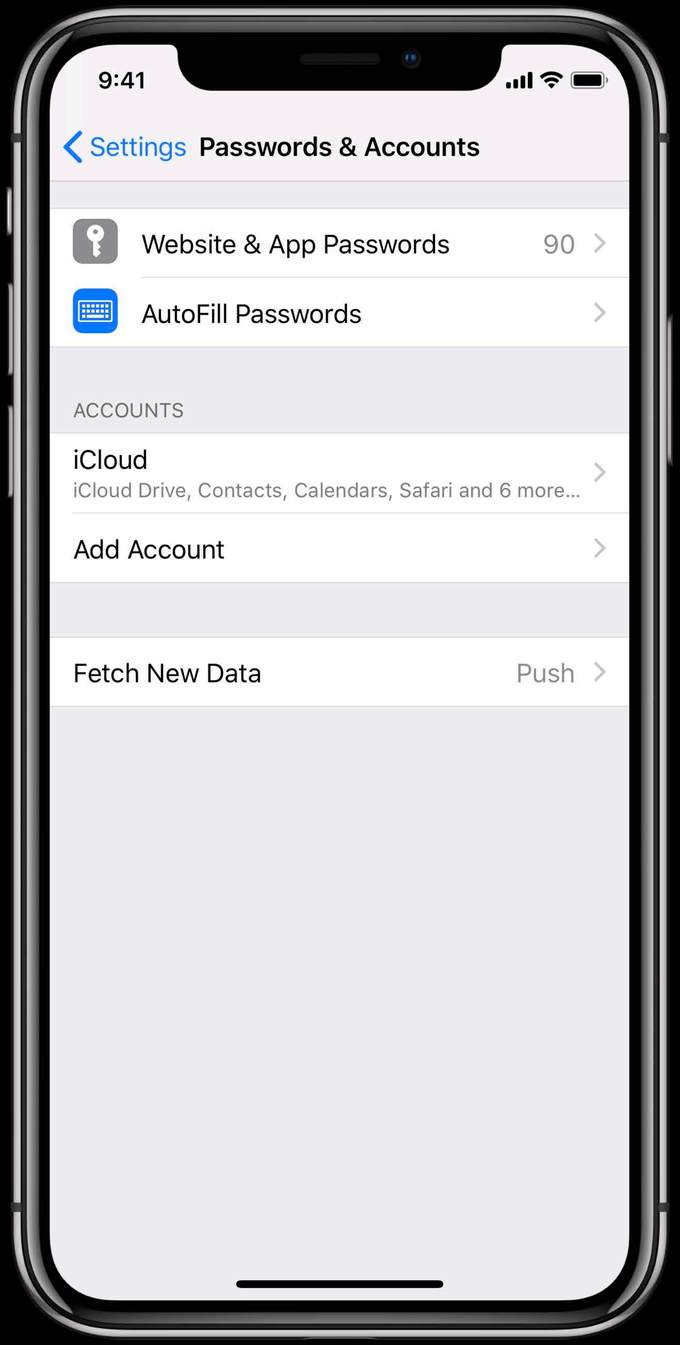

9:41
example.com
shiny.
Username
Password
Log In
Passwords
Done
password for this website
j_appleseed
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
return

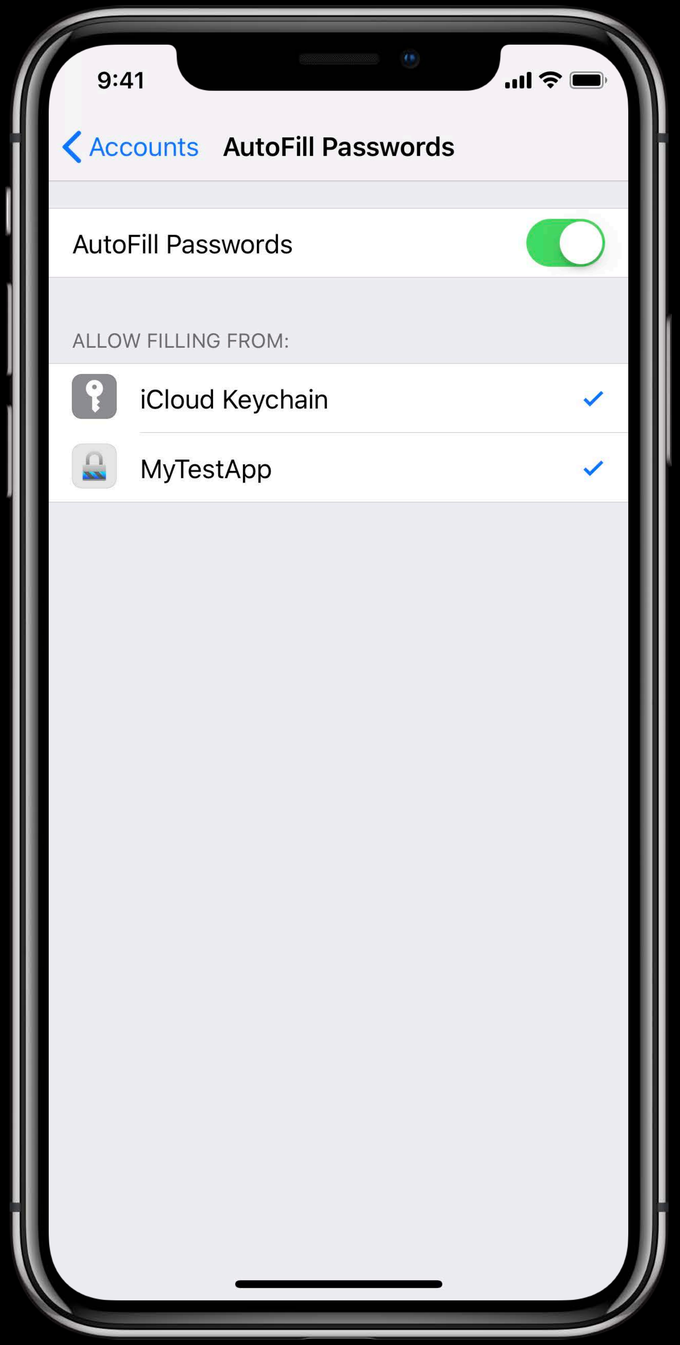

9:41
shiny
Email
Password
Log In
password for example.com
j_appleseed
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return


AutoFill

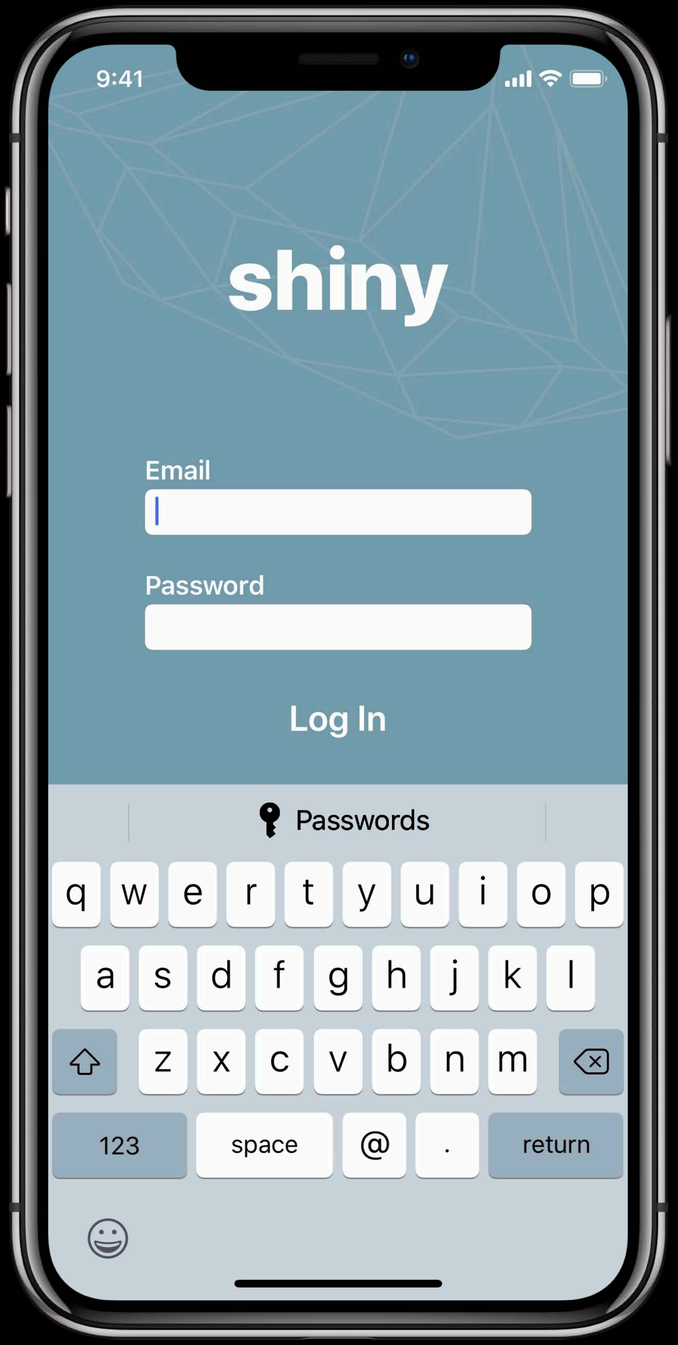

AutoFill

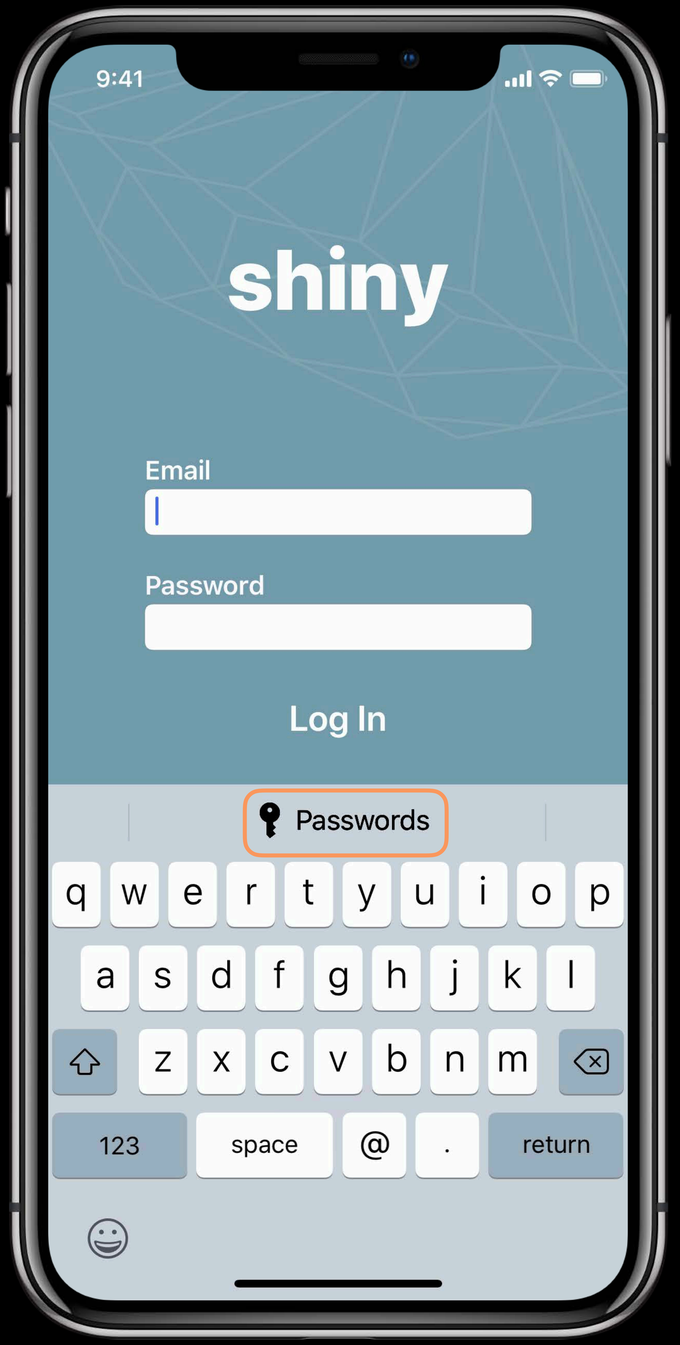

AutoFill
tv

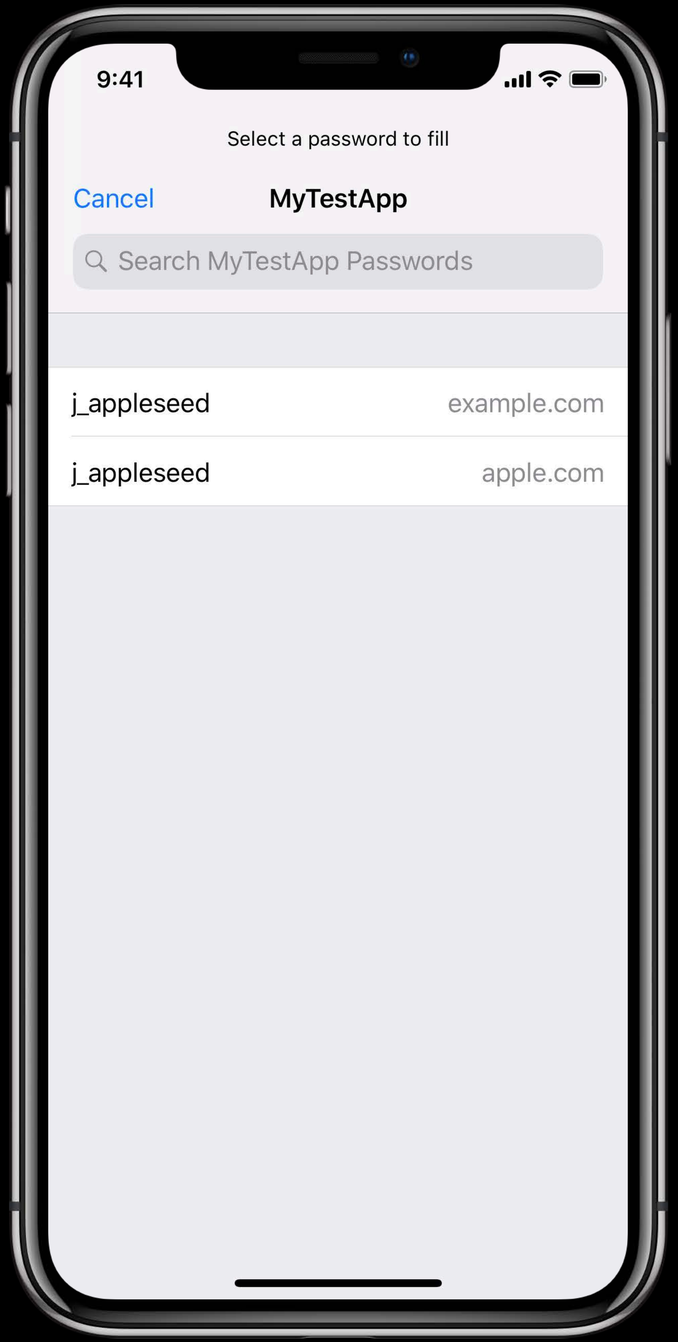

9:41
Settings
Passwords & Accounts
Website & App Passwords
90
AutoFill Passwords
ACCOUNTS
iCloud
iCloud Drive, Contacts, Calendars, Safari and 6 more...
Add Account
Fetch New Data
Push


9:41
Accounts
AutoFill Passwords
AutoFill Passwords
ALLOW FILLING FROM:
iCloud Keychain
MyTestApp

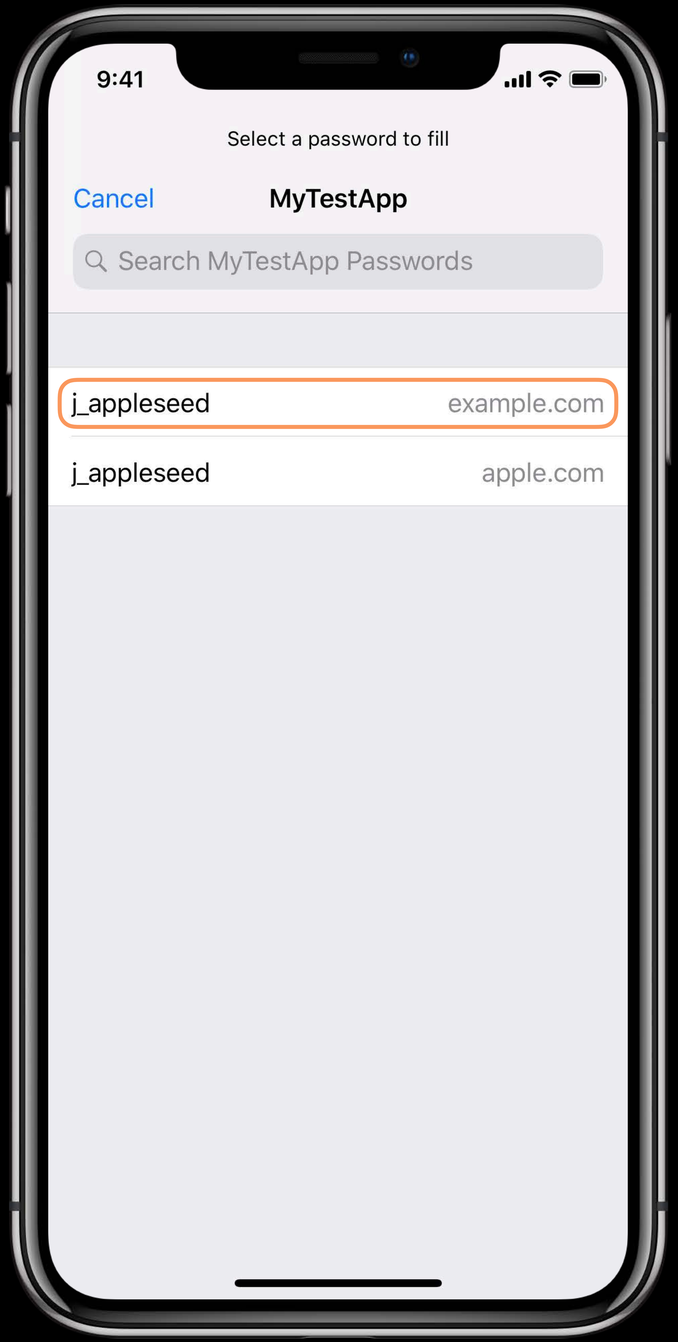
NEW


9:41
shiny
Email
Password
Log In
Passwords
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return


9:41
shiny
Email
Password
Log In
Passwords
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return


9:41
Select a password to fill
Cancel
MyTestApp
Search MyTestApp Passwords
j_appleseed
example.com
j_appleseed
apple.com

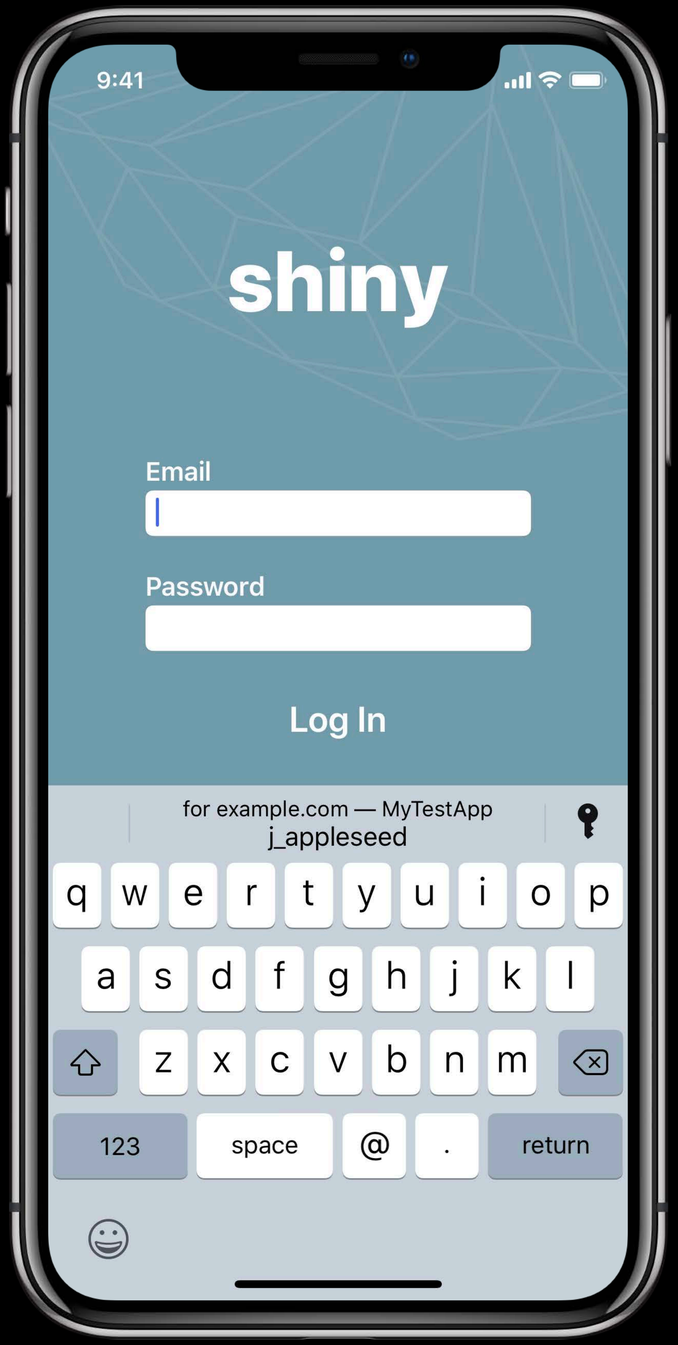
NEW


9:41
Select a password to fill
Cancel
MyTestApp
Search MyTestApp Passwords
j_appleseed
example.com
j_appleseed
apple.com

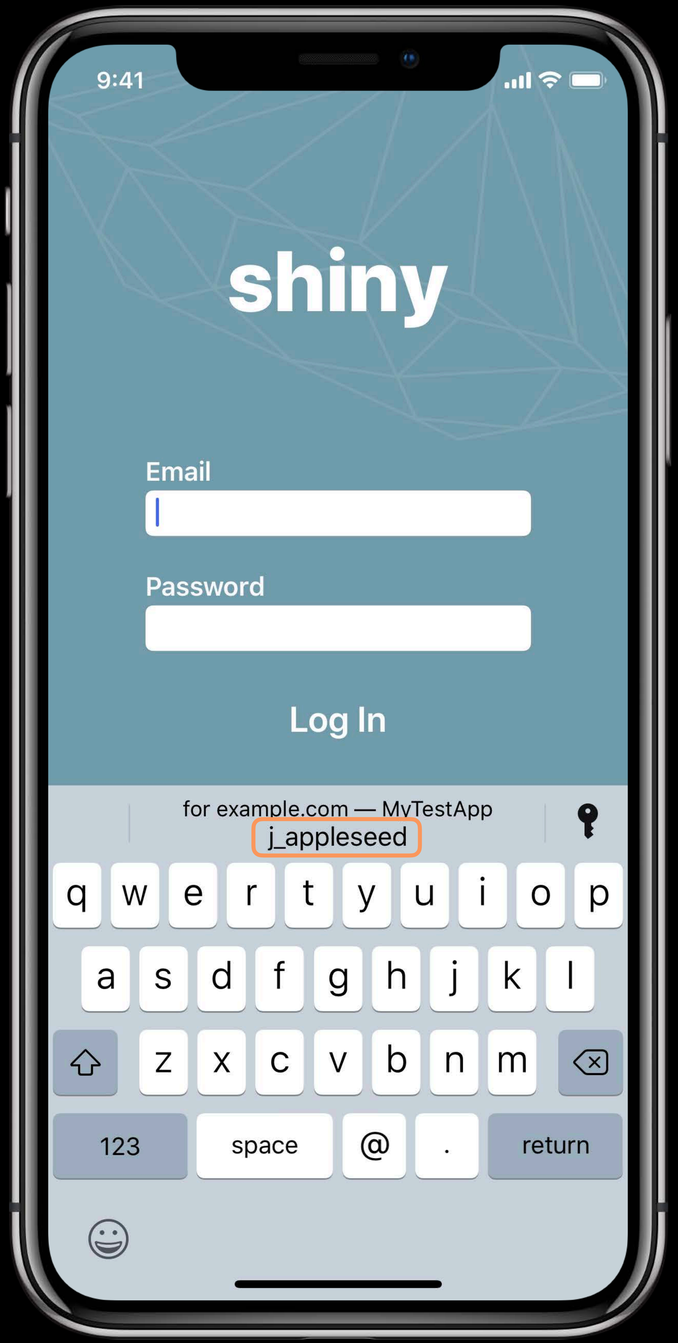
NEW


9:41
shiny
Email
j_appleseed
Password
Log In
Forgot Password?
Create Account

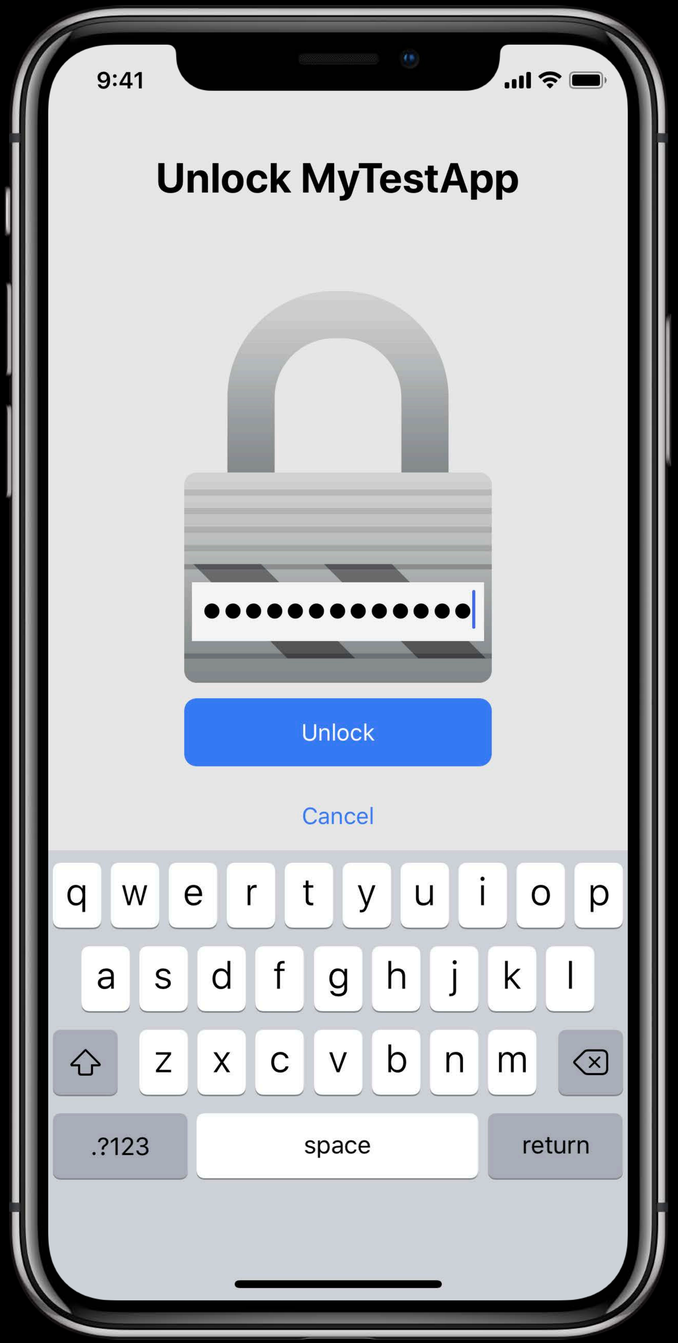
NEW


9:41
shiny
Email
Password
Log In
for example.com — MyTestApp
j_appleseed
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return

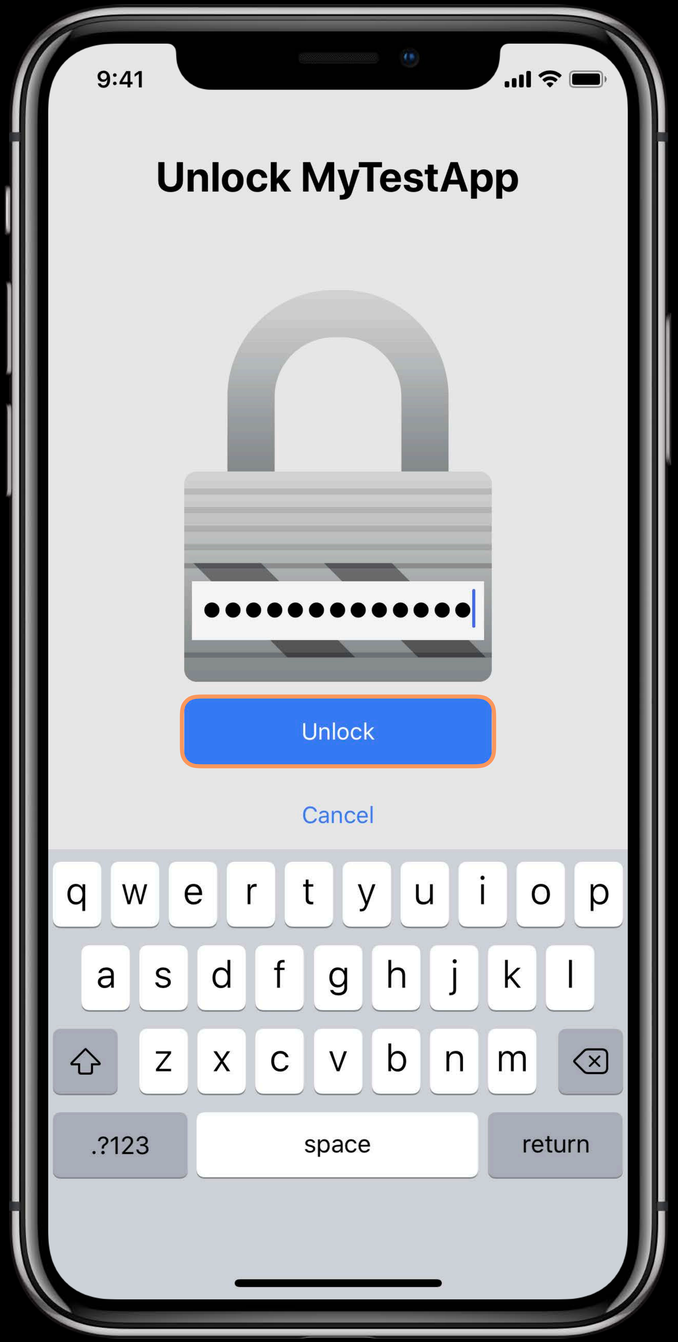
NEW


9:41
shiny
Email
Password
Log In
for example.com — MyTestApp
j_appleseed
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return

NEW


9:41
Unlock MyTestApp
Unlock
Cancel
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
.?123
space
return

NEW


9:41
Unlock MyTestApp
Unlock
Cancel
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
.?123
space
return

NEW


9:41
shiny
Email
j_appleseed
Password
Log In
Forgot Password?
Create Account

NEW

# 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

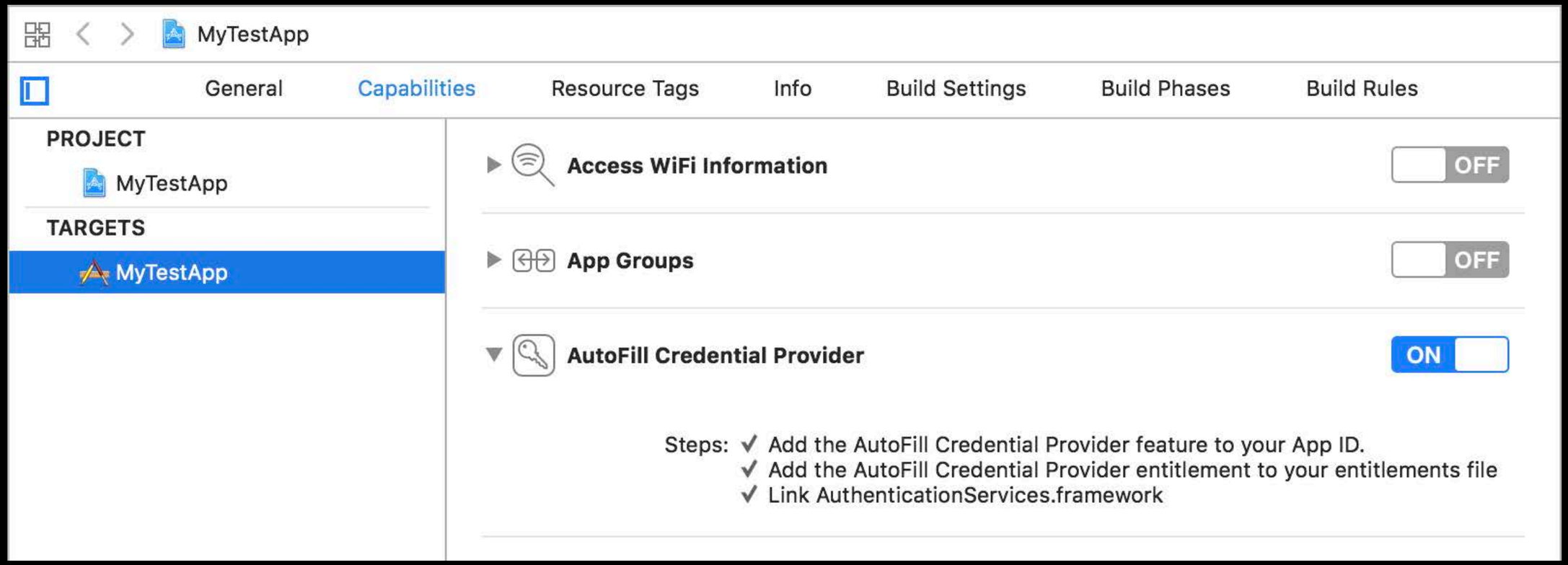
Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings

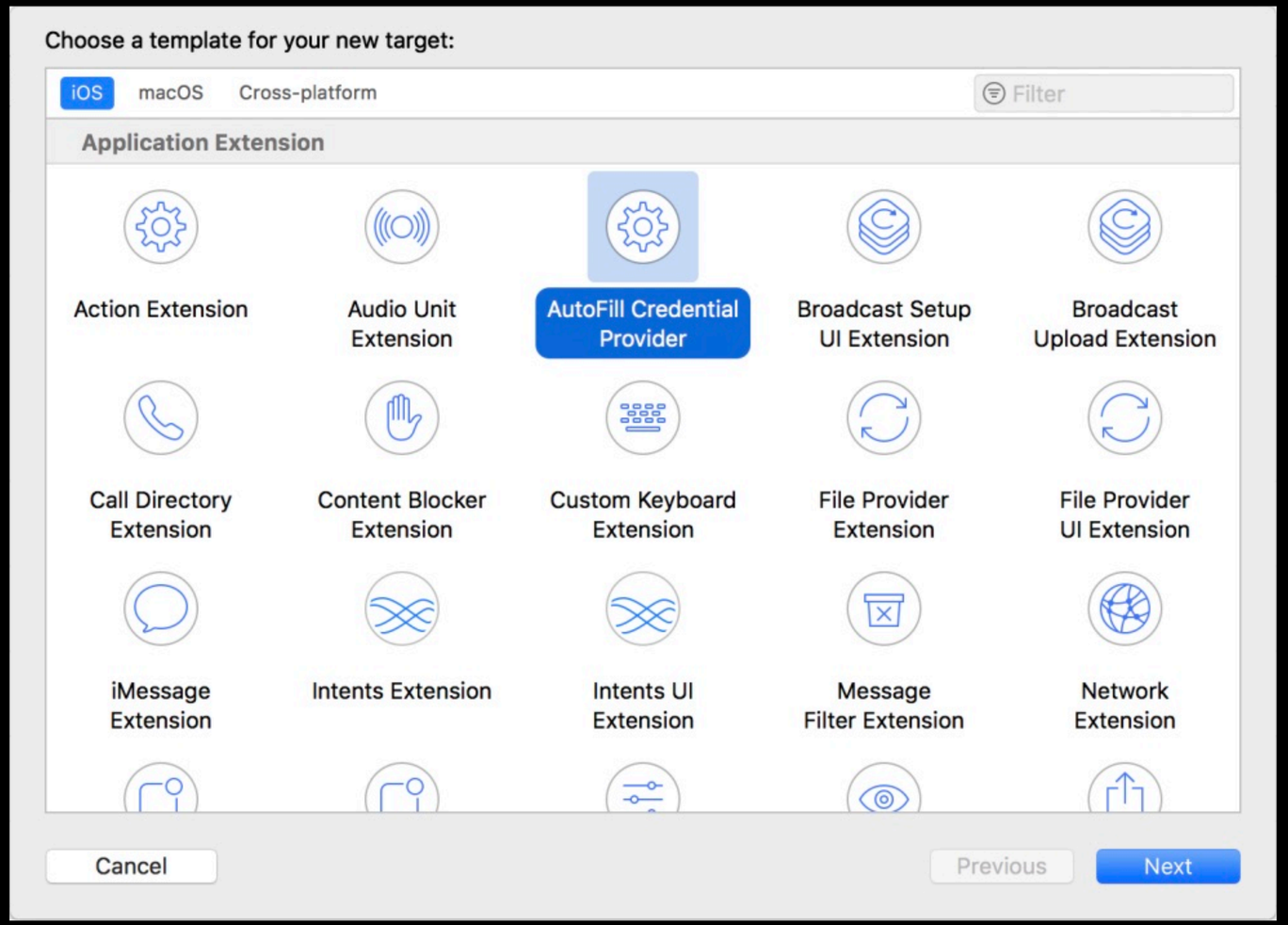
# Configure Your Project

## Enable the ‘AutoFill Credential Provider’ capability



# 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

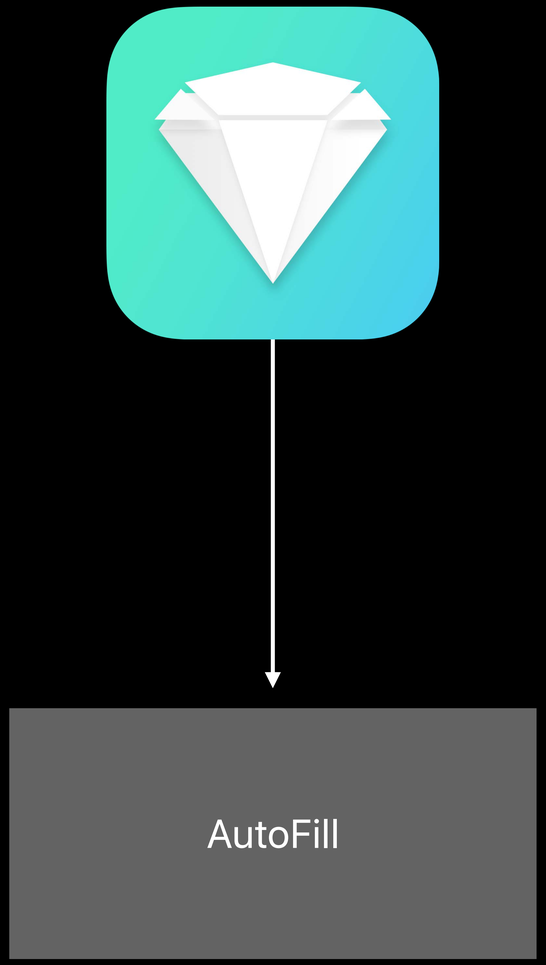
Implement a credential list

Show credentials in the QuickType bar

Allow users to configure your extension in Settings

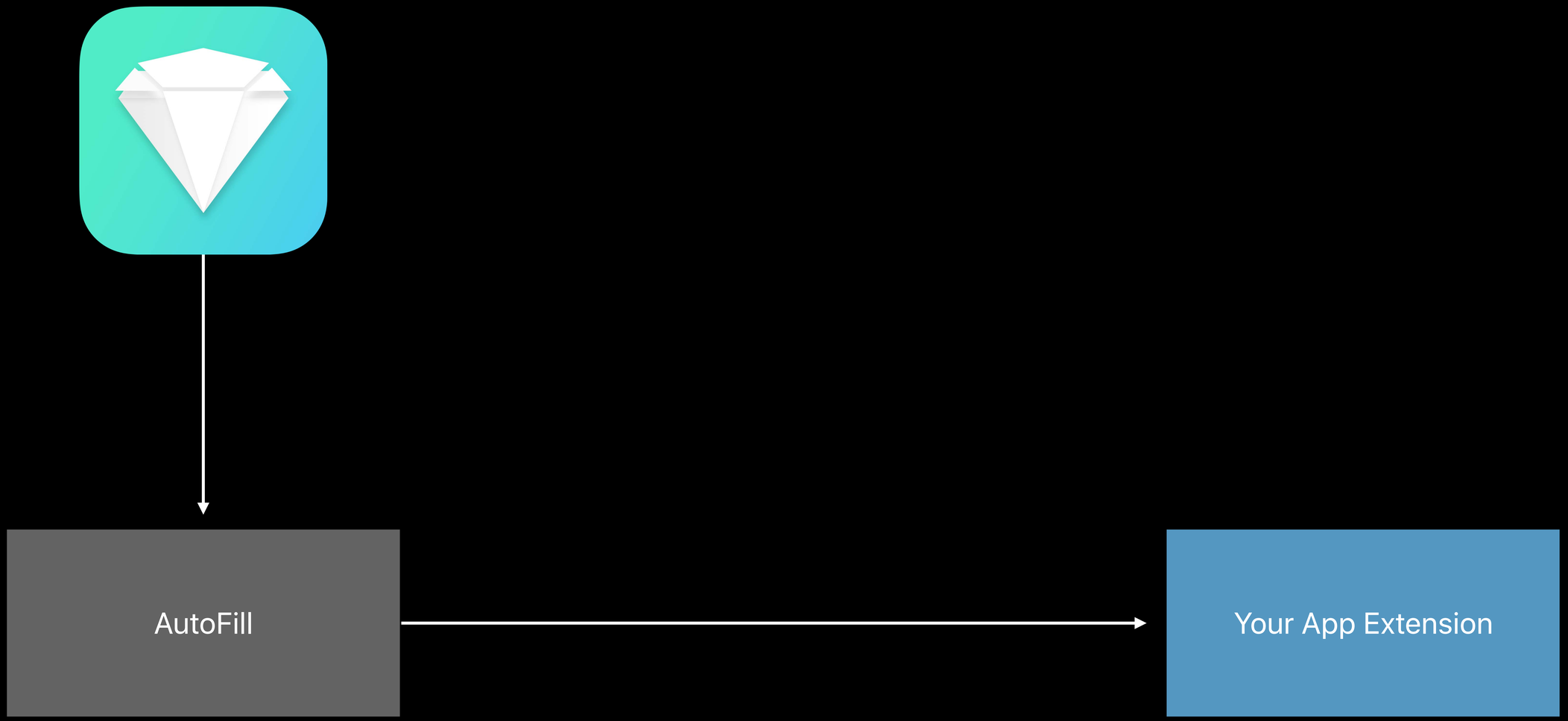
# Implement a Credential List

## Overview



# Implement a Credential List

## Overview



# 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

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

ASCredentialServiceIdentifier

AutoFill

Your App Extension

# Implement a Credential List

## Overview

```
class ASCredentialProviderViewController
    : UIViewController {

  func prepareCredentialList(for serviceIdentifiers:
          [ASCredentialServiceIdentifier]) {

    // Show the credential list UI

  }

}
```

# Implement a Credential List

## Overview

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

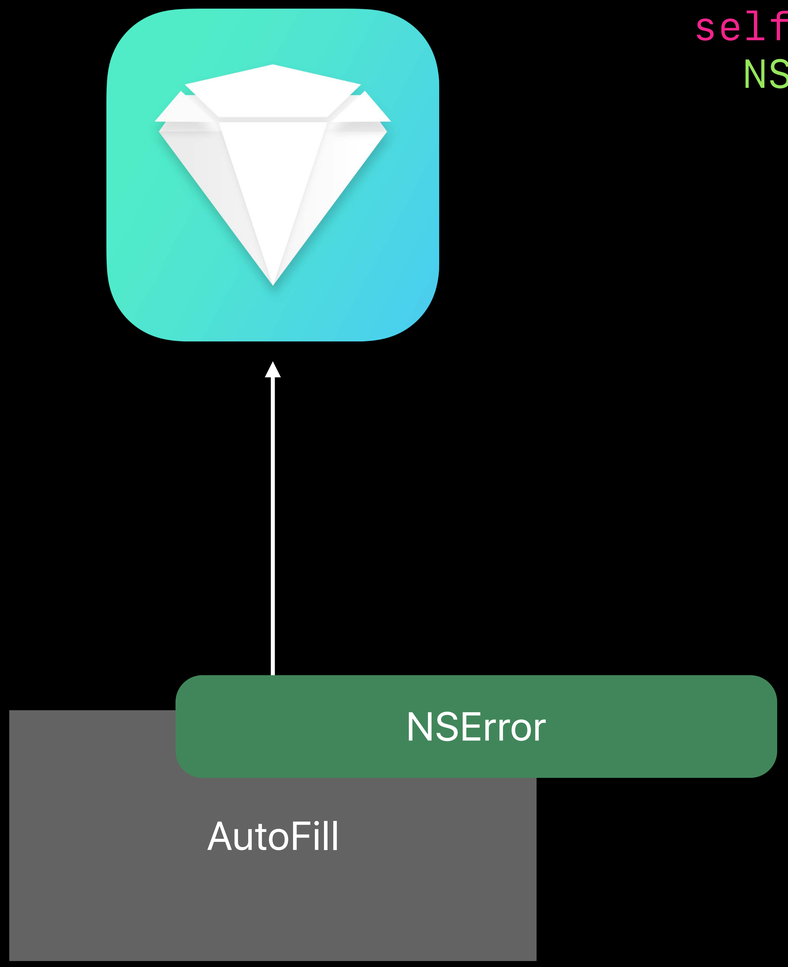
# Implement a Credential List

## Overview

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

# Implement a Credential List

## Overview




```
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)
```

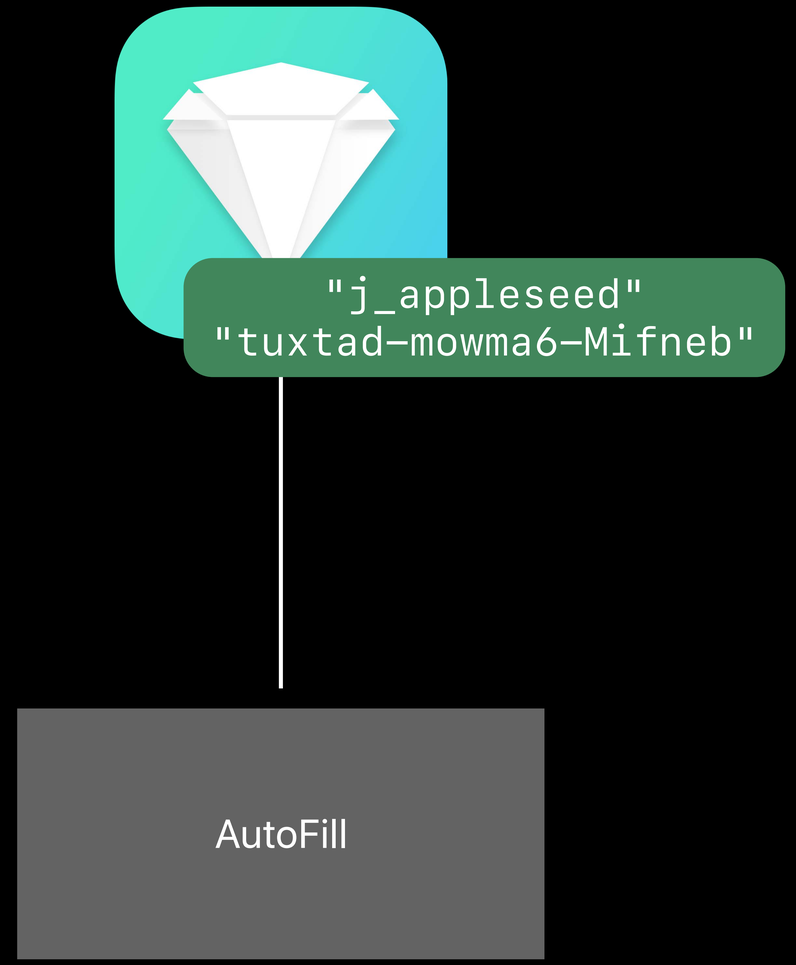
# Implement a Credential List

## Overview

```
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



# 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

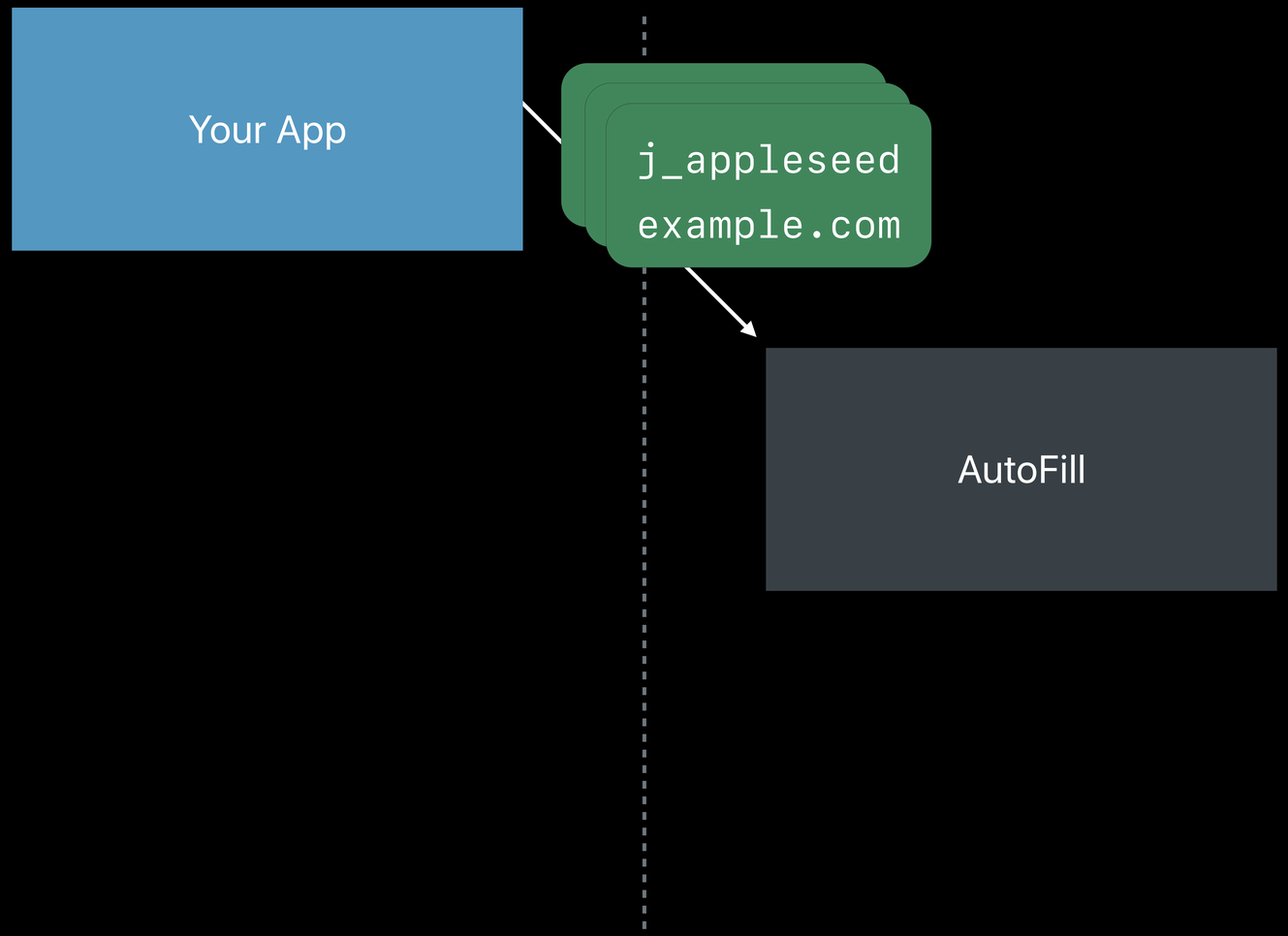
Allow users to configure your extension in Settings

# Show Credentials in the QuickType Bar

Overview

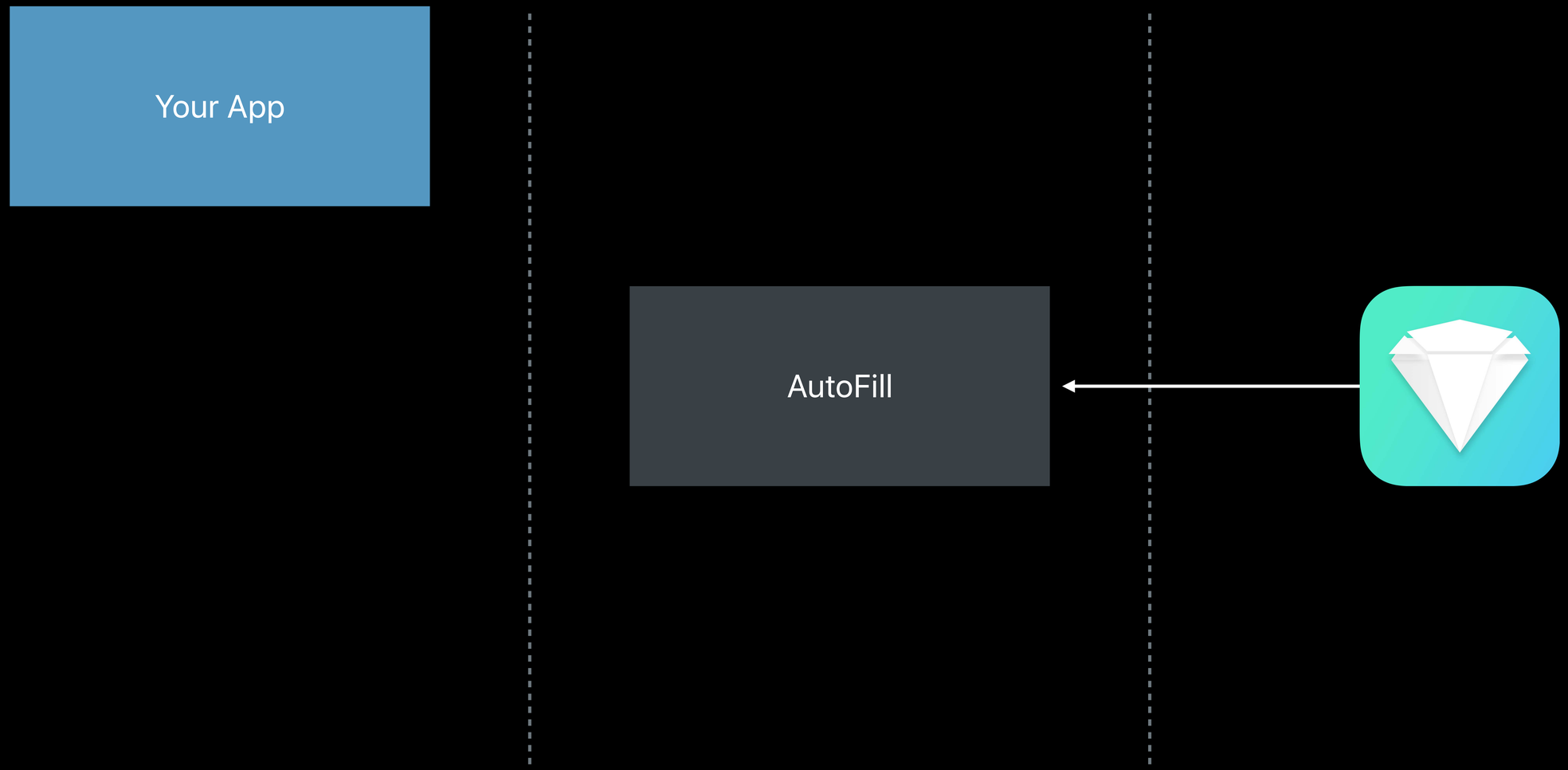
# Show Credentials in the QuickType Bar

## Overview



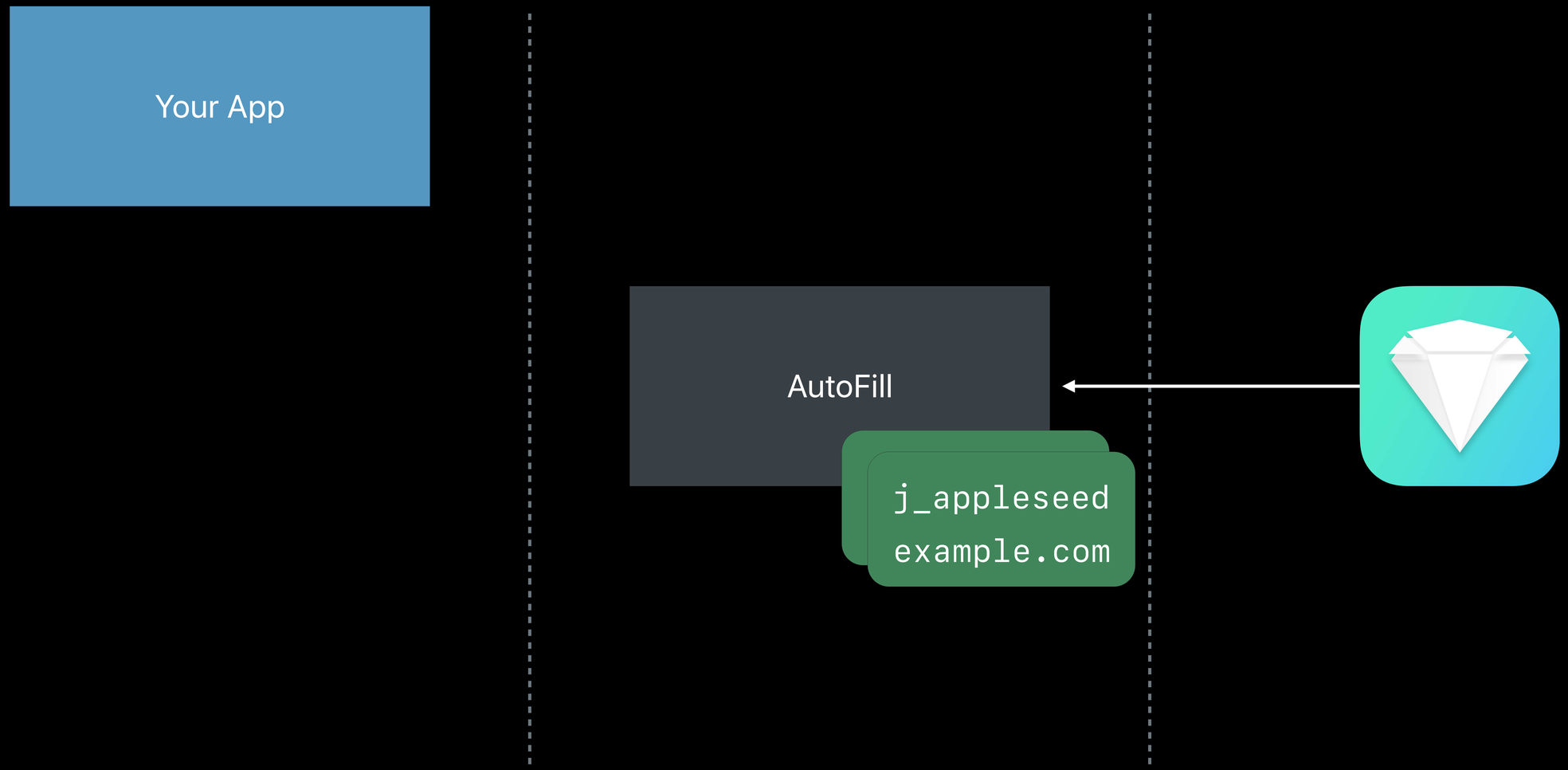
# Show Credentials in the QuickType Bar

## Overview



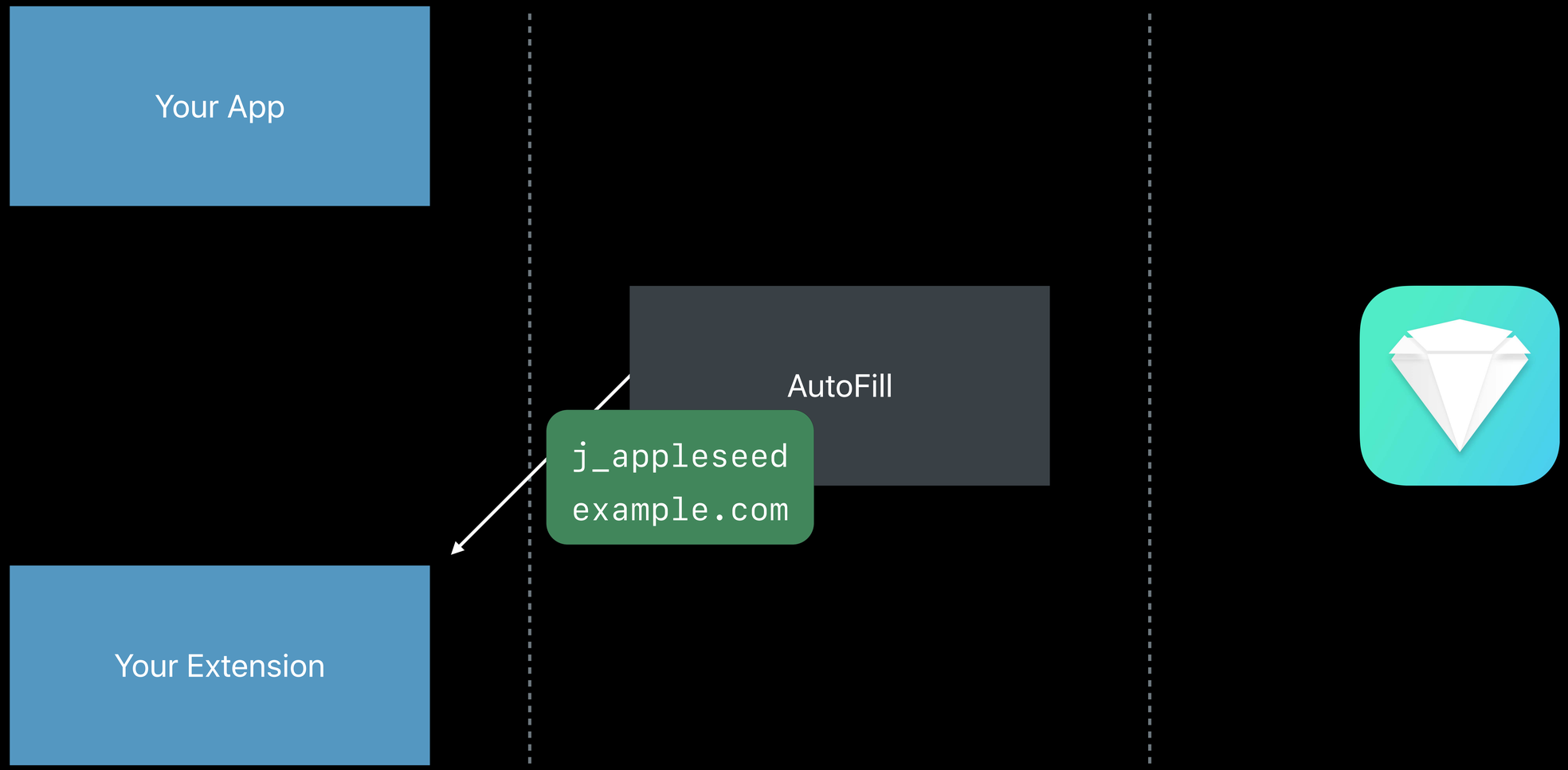
# Show Credentials in the QuickType Bar

## Overview



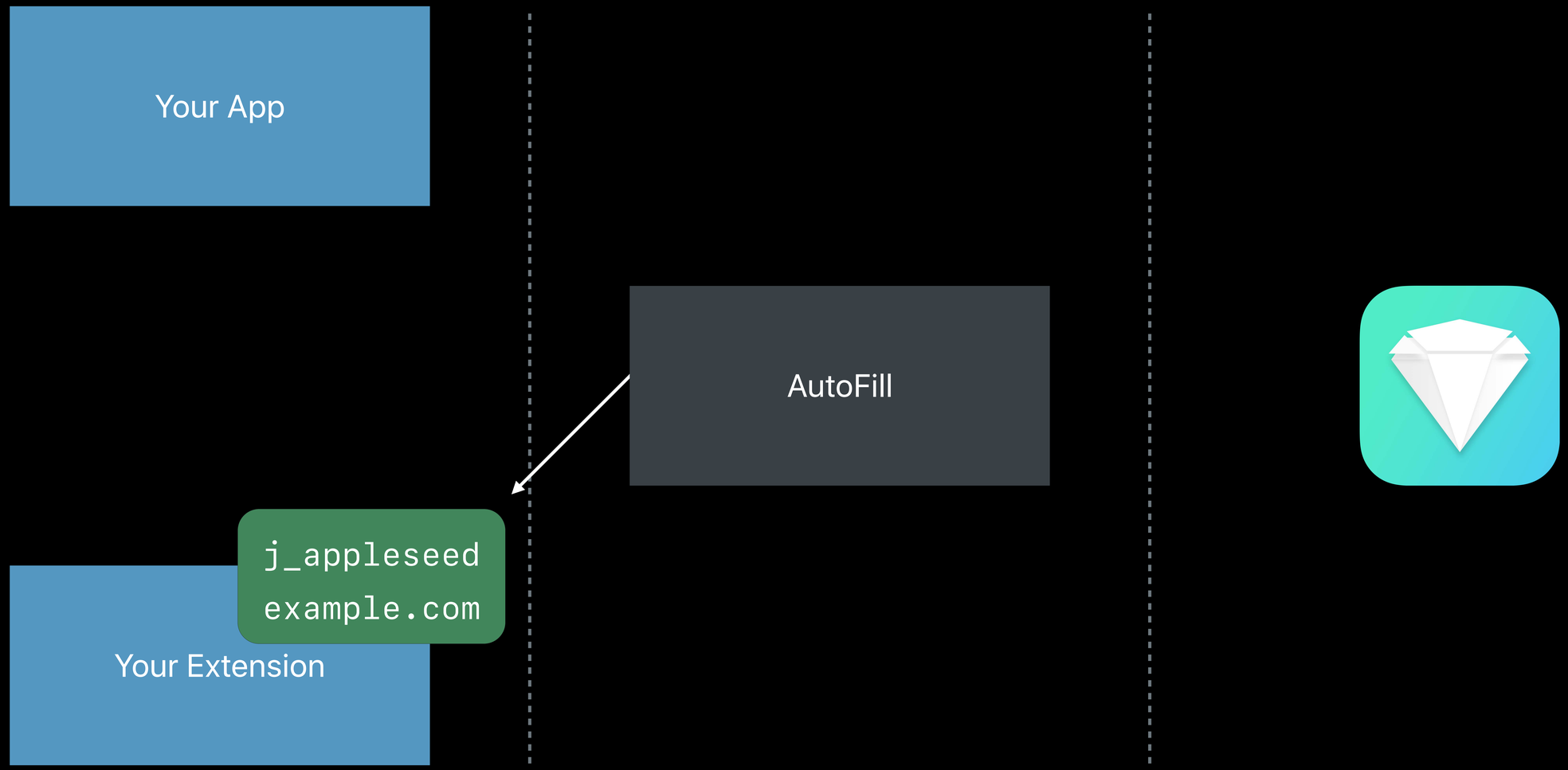
# Show Credentials in the QuickType Bar

## Overview



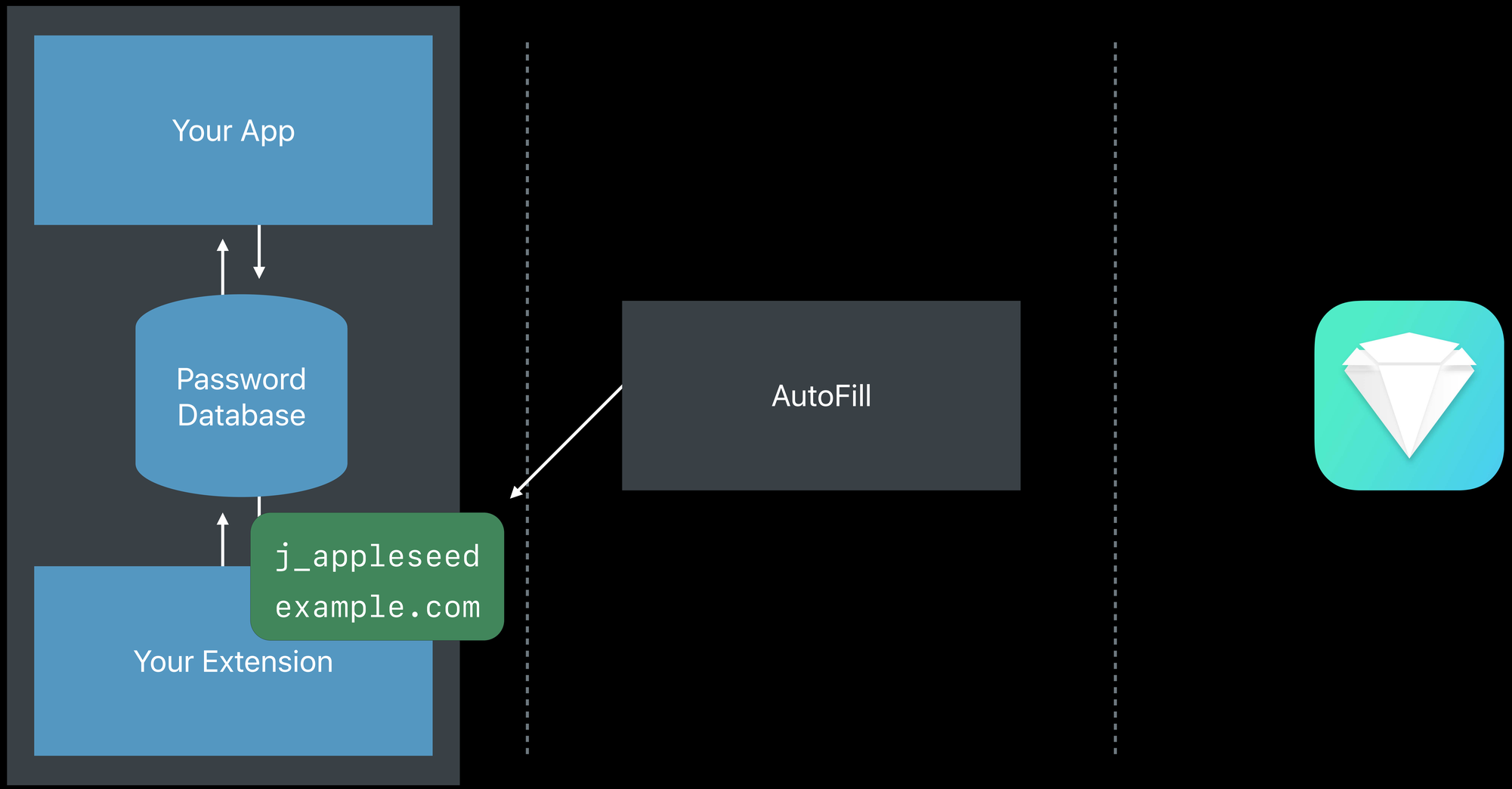
# Show Credentials in the QuickType Bar

## Overview



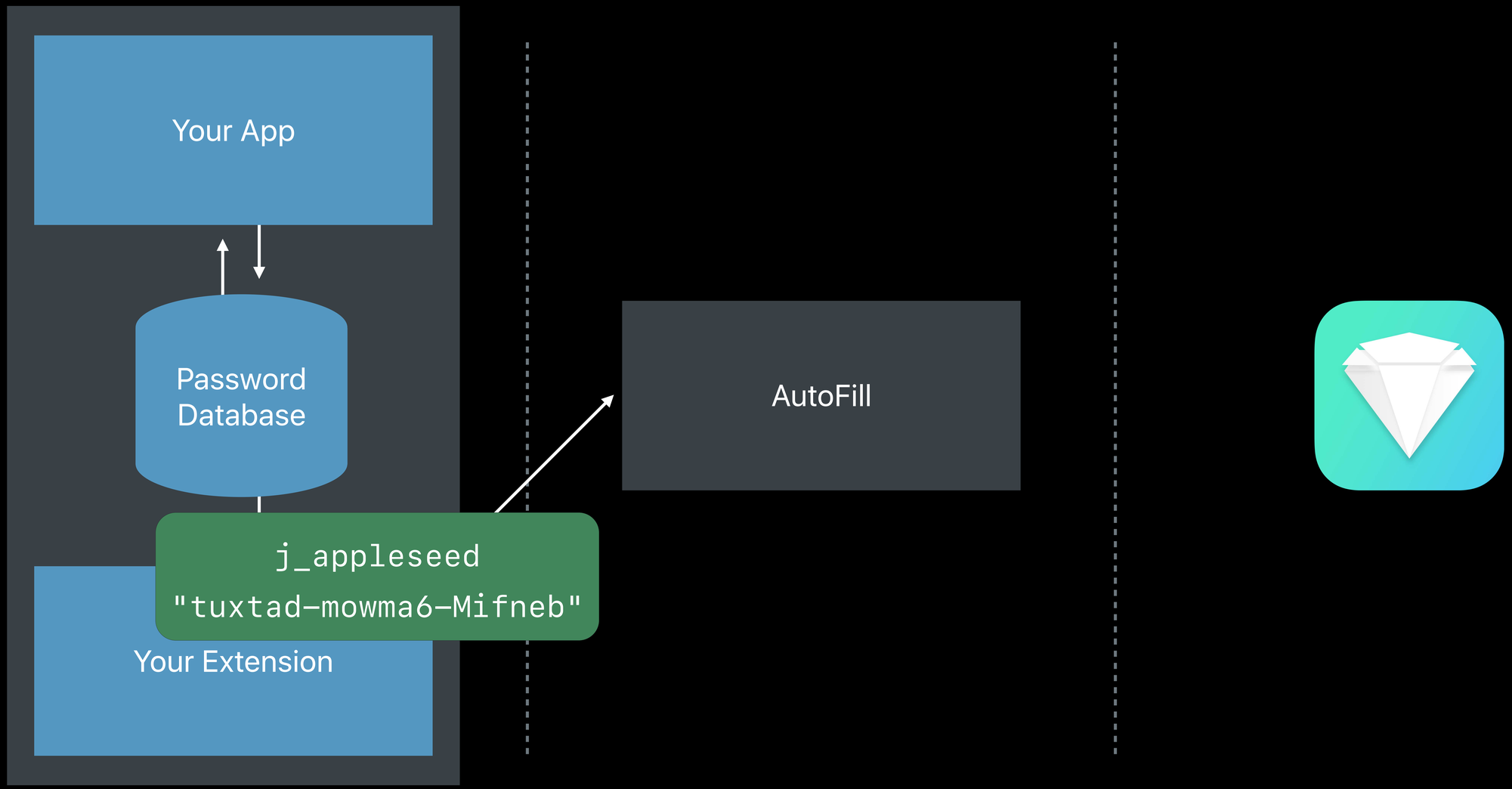
# Show Credentials in the QuickType Bar

## Overview



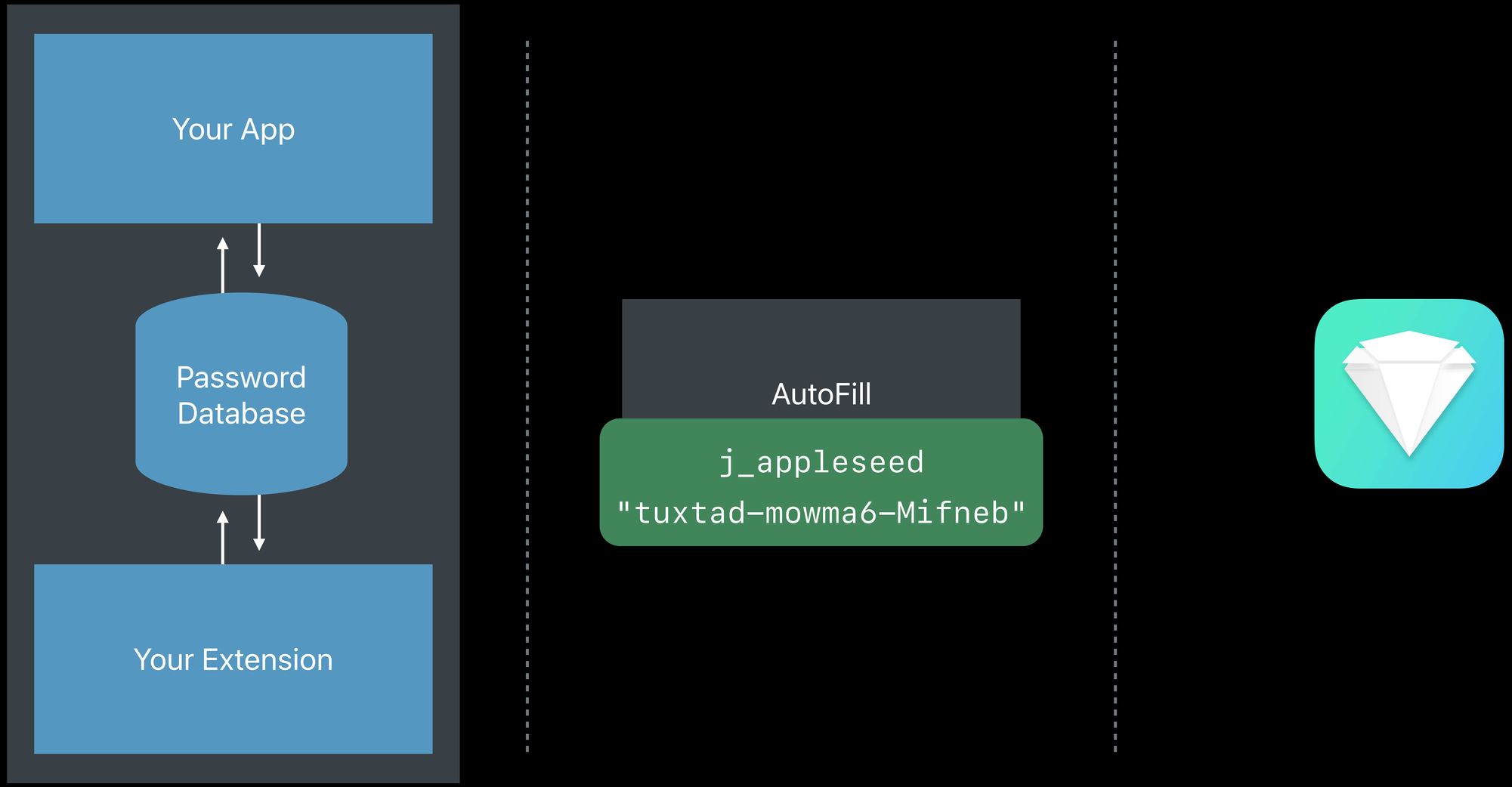
# Show Credentials in the QuickType Bar

## Overview



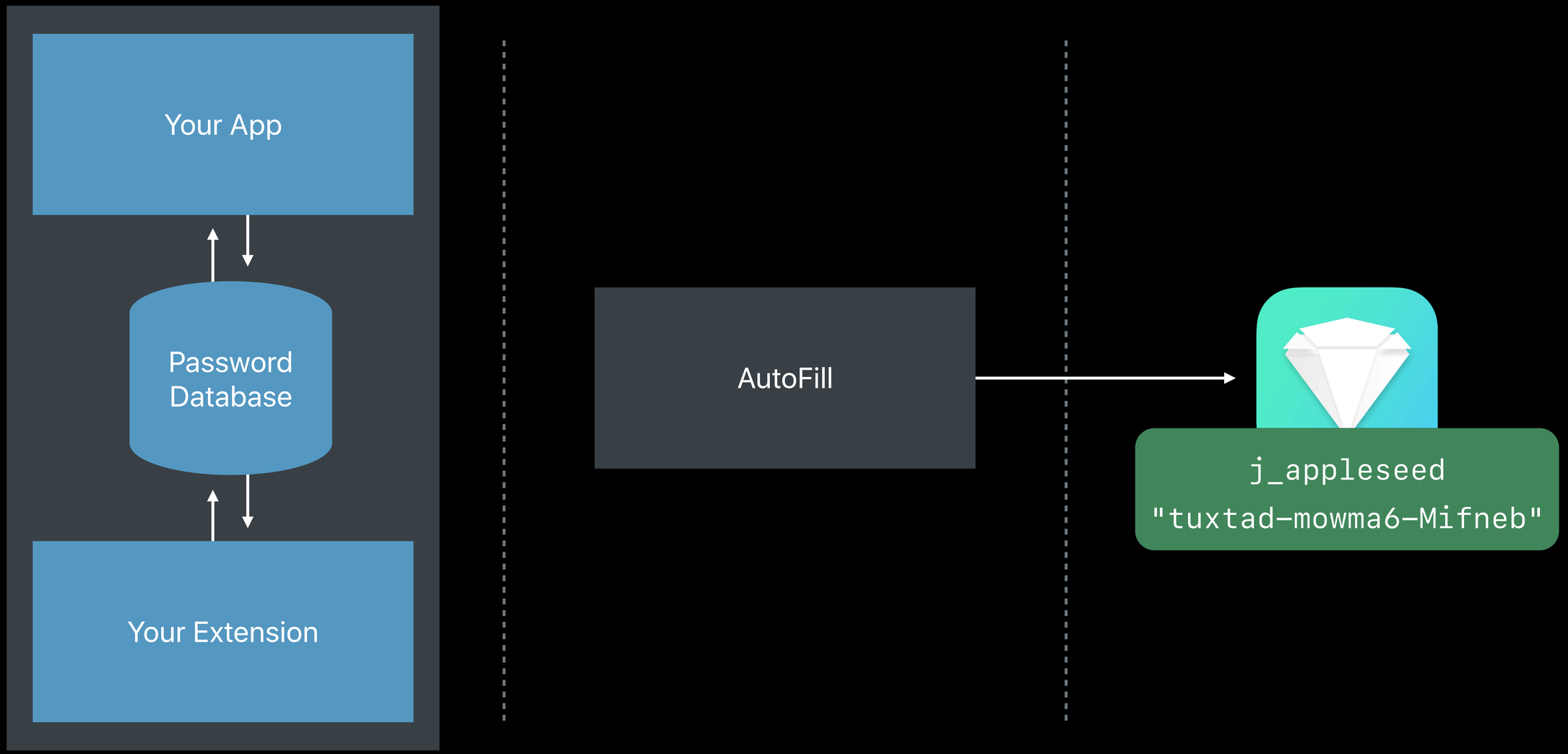
# Show Credentials in the QuickType Bar

## Overview



# 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

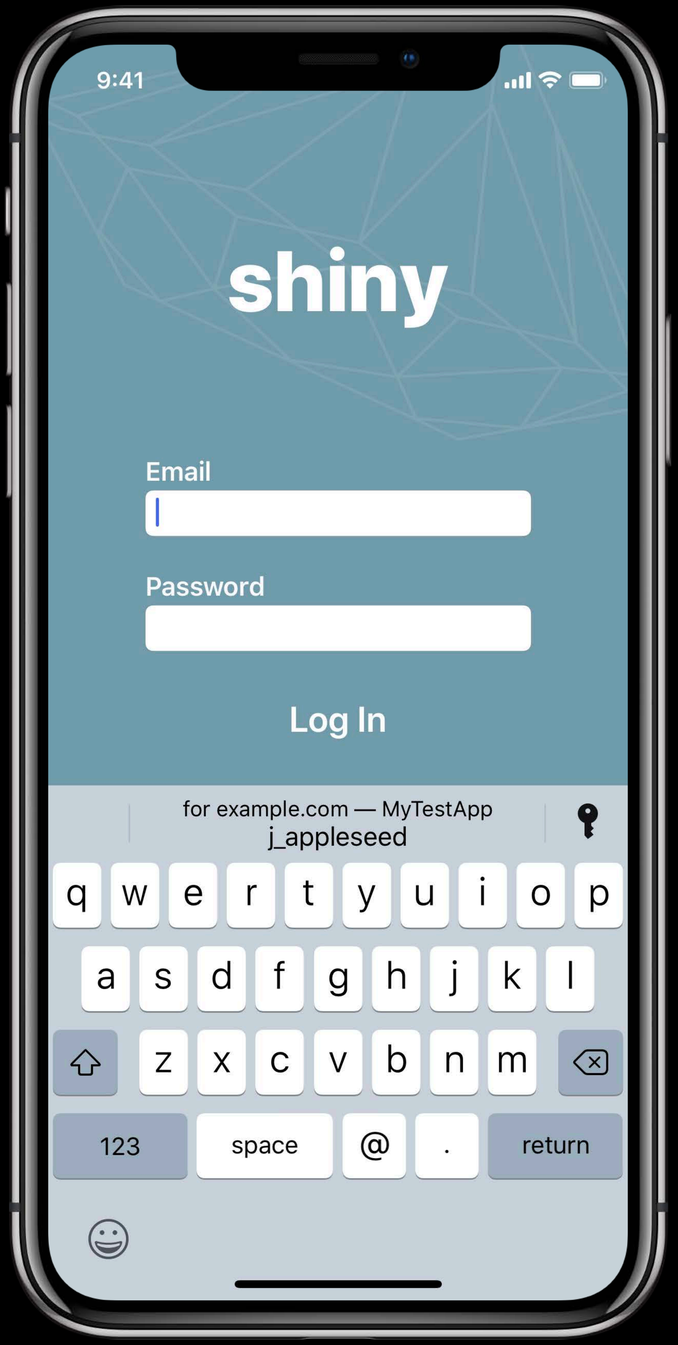
ASCredentialIdentityStore can help

```
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 }

}
```


9:41
shiny
Email
Password
Log In
for example.com — MyTestApp
j_appleseed
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
@
.
return

```
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

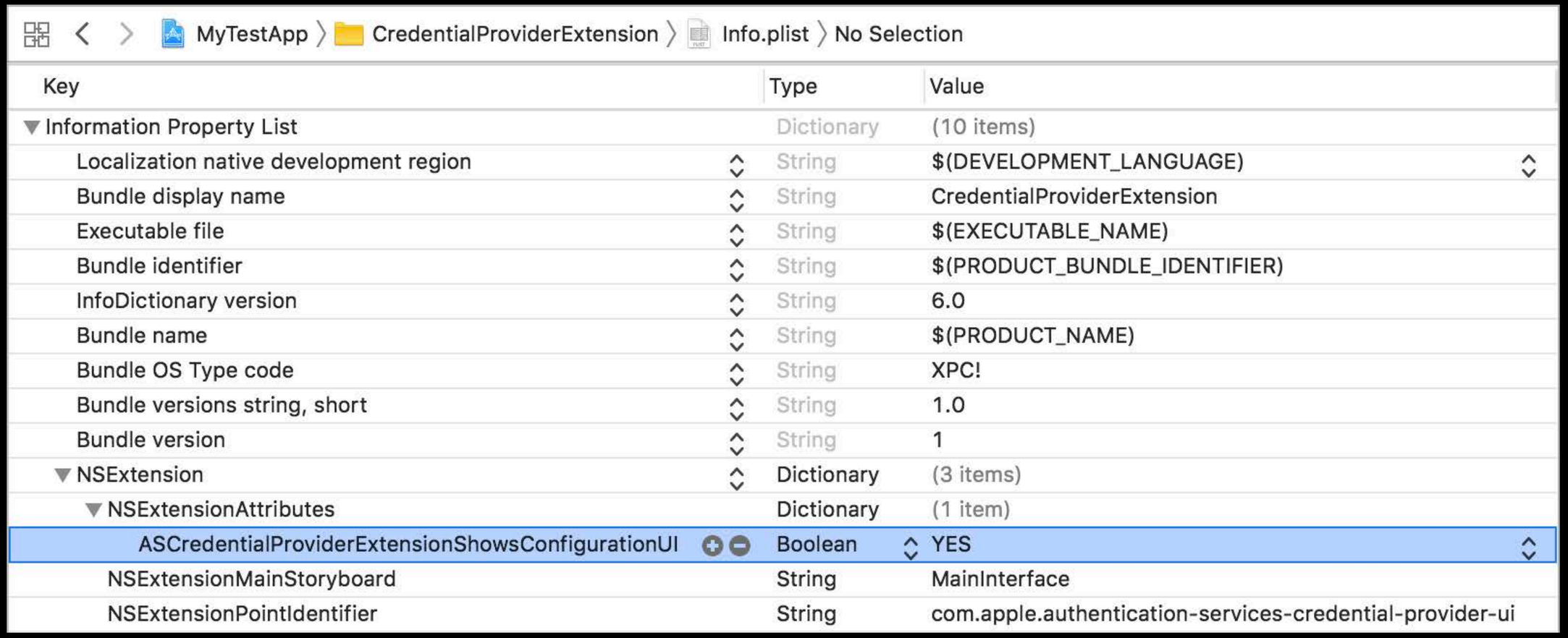
# Allow Users to Configure Your Extension in Settings

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



```
class CredentialProviderViewController: ASCredentialProviderViewController {

    func prepareInterfaceForExtensionConfiguration() {

        // Present appropriate Settings UI

    }









}
```

```swift
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

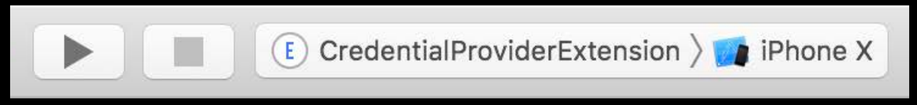
Use Safari to test filling credentials

# Extension Development

## Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials

- Select your extension’s scheme

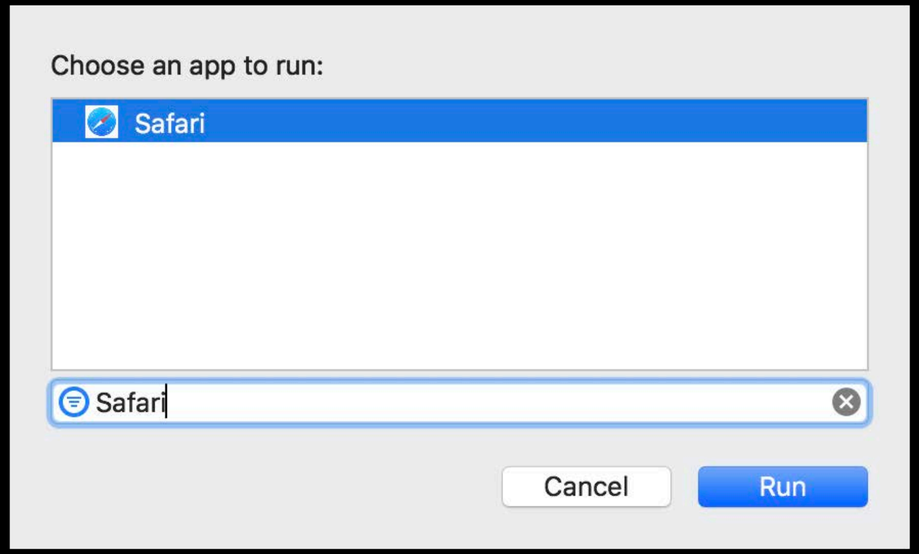


# Extension Development

## Debugging AutoFill Credential Provider extensions

Use Safari to test filling credentials

- Select your extension’s scheme
- Run with Safari as a target

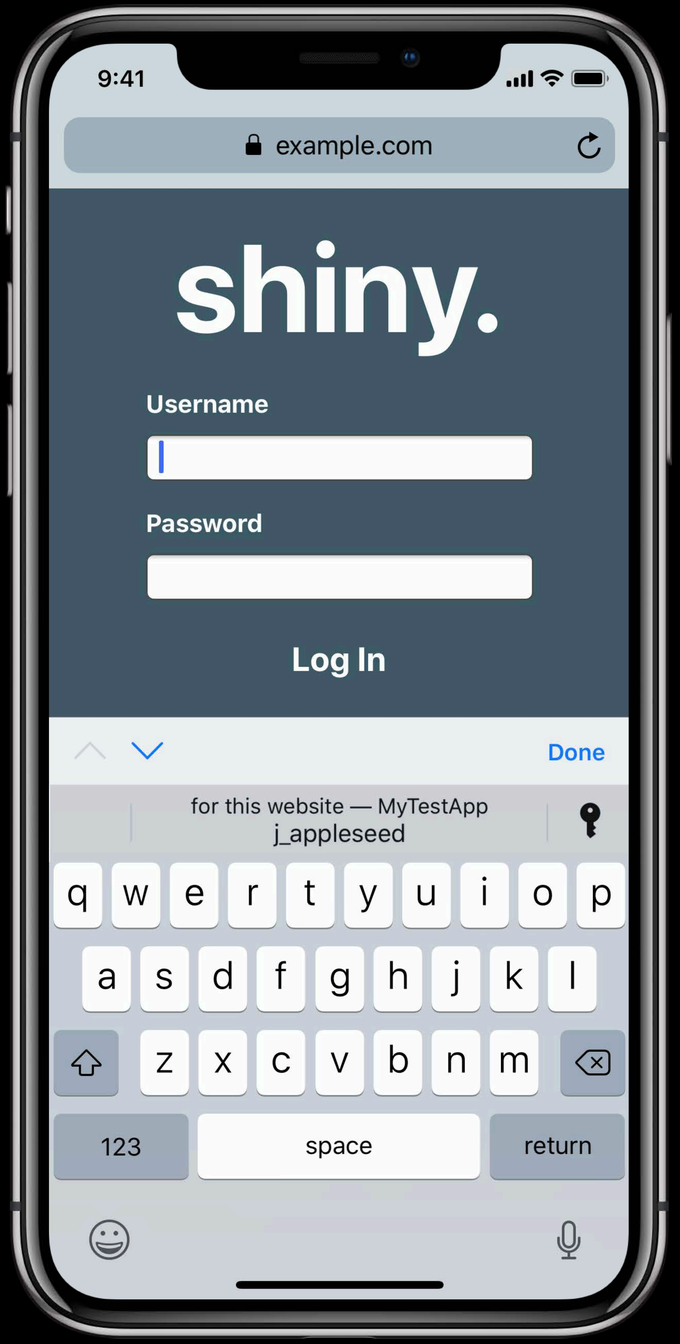


# Extension Development

## 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 | WWDC18 |
| Introducing Password AutoFill for Apps | WWDC17 |


WWDC18