Sample Code

Mapping Apple TV users to app profiles

Adapt the content of your app for the current viewer by using an entitlement and simplifying sign-in flows.

Download

tvOS 15.0+ | Xcode 14.0+

Overview

This sample demonstrates how to reduce configuration dialogs for people who share an Apple TV and a service account, such as a family media subscription. To let people get to your content more directly, your app should save shared login information where it can access the credentials regardless of the current user, and your code should also remember which profile to load for each user.

> **Note**
>
> This sample code project is associated with WWDC22 session 110384: Support multiple users in tvOS apps.

Configure the sample code project

Before you run the sample code project in Xcode:

- Update your Apple TV to tvOS 15 or later. This app runs in single-user mode on tvOS 15 and supports multiple users on tvOS 16 and later.
- Add users by choosing Settings > Users and Accounts > Add New User.

- To switch TV users, press and hold the TV button on your remote or choose Settings > Users and Accounts > Switch Current User.

Run as the current user

Apple TV supports multiple users. To opt-in to keeping separate data for each user, add the User Management Entitlement to your app or app extension, with the value `runs-as-current-user-with-user-independent-keychain`.

```
<key>com.apple.developer.user-management</key>
<array>
    <string>runs-as-current-user-with-user-independent-keychain</string>
</array>
```

> **Note**
>
> This entitlement value grants your app per-user data storage on tvOS 16 and higher but maintains single-user behavior on tvOS 15 or earlier.

Save each user's profile

When your app or extension is running on tvOS 16 with multiple Apple TV users, you can store each person's preferred profile directly with an API such as `UserDefaults` or CloudKit; the system separates the data for each Apple TV user. Your code doesn't have to handle any data partitioning.

Here's an example using `UserDefaults` to save the profile.

```
private var selectedProfileIdentifier: String? {
    didSet {
        guard oldValue != selectedProfileIdentifier else { return }

        // If running on tvOS 16, check if the selected profile should be
        // remembered before storing it.
        if #available(tvOS 16.0, *), userManager.shouldStorePreferencesForCurrentUse
            UserDefaults.standard.set(selectedProfileIdentifier, forKey: "PreferredF
        }
```

The sample only attempts to save the selected profile when running on tvOS 16 or later; it doesn't run as the current user on tvOS 15 or earlier. It also checks `shouldStorePreferencesFor`

`CurrentUser` before saving to avoid remembering the selected profile in case there aren't multiple users on the Apple TV.

Skip the profile-picker dialog

This sample app can skip the interstitial profile-picker dialog whenever a person who already has a saved profile launches the app.

```
private func presentProfilePickerIfNeeded() {
    if #available(tvOS 16.0, *) {
        // On tvOS 16, present the profile picker only if the Apple TV
        // doesn't have multiple users, or the current user hasn't picked a
        // profile yet.
        if !userManager.shouldStorePreferencesForCurrentUser || profileData.selected
            presentProfilePicker()
        }
    } else {
        // Always present the profile picker on tvOS 15 and earlier.
        presentProfilePicker()
    }
}
```

The sample still presents the profile picker if the Apple TV doesn't have multiple users, the current user doesn't have a saved profile, or the app is running on tvOS 15 or earlier.

Store shared login credentials in the user-independent Keychain

Another interstitial dialog that can get in the way of the content is the sign-in screen. Keychain data is automatically separated for each Apple TV user. To avoid having each user sign in individually, tvOS 16 adds a new Keychain API: `kSecUseUserIndependentKeychain`.

With `kSecUseUserIndependentKeychain`, signing in only needs to happen once, even when running as the current user, because items stored using this property are accessible by all Apple TV users.

```
var baseQuery: [CFString: Any] = [
    kSecAttrService: "com.example.apple-samplecode.ProfilesSample",
    kSecClass: kSecClassGenericPassword
]
if #available(tvOS 16.0, *) {
```

```
    baseQuery[kSecUseUserIndependentKeychain] = kCFBooleanTrue as AnyObject
}
self.baseQuery = baseQuery
```

The `loadCredentials` method in the `KeychainController` shows how to use the `baseQuery` above to read the username and password from the Keychain:

```
private func loadCredentials() {
    var attributesQuery = baseQuery
    attributesQuery[kSecReturnAttributes] = kCFBooleanTrue

    // Read all attributes. This is where the username comes from.
    var outAttributes: AnyObject?
    guard SecItemCopyMatching(attributesQuery as CFDictionary, &outAttributes) == er
          let attributes = outAttributes as? [CFString: Any] else {
        return
    }

    // This is where the password comes from.
    var passwordQuery = baseQuery
    passwordQuery[kSecAttrAccount] = attributes[kSecAttrAccount]
    passwordQuery[kSecReturnData] = kCFBooleanTrue

    var outPassword: AnyObject?
    guard SecItemCopyMatching(passwordQuery as CFDictionary, &outPassword) == errSec
          let passwordData = outPassword as? Data else {
        return
    }

    if let username = attributes[kSecAttrAccount] as? String,
        let password = String(data: passwordData, encoding: .utf8) {
        credentials = (username: username, password: password)
    }
}
```

You can also use the same `baseQuery` to save a new item with username and password.

```
func save(username: String, password: String) {
    guard let passwordData = password.data(using: .utf8) else {
        return
    }
```

```
    let attributes: [CFString: Any] = [
        kSecAttrAccount: username,
        kSecValueData: passwordData
    ]

    var status: OSStatus = errSecCoreFoundationUnknown
    var itemExists = SecItemCopyMatching(baseQuery as CFDictionary, nil) == errSecSu

    // Try to add the item to the keychain first.
    if !itemExists {
        let addAttributes = baseQuery.merging(attributes) { (current, _) in current
        status = SecItemAdd(addAttributes as CFDictionary, nil)
        itemExists = status == errSecDuplicateItem
    }

    // Otherwise, update if it already exists.
    if itemExists {
        status = SecItemUpdate(baseQuery as CFDictionary, attributes as CFDictionary
    }

    guard status == errSecSuccess else {
        return
    }

    credentials = (username, password)
}

func removeCredentials() {
    SecItemDelete(baseQuery as CFDictionary)
    credentials = nil
}
```

See Also

Multiple users

Personalizing Your App for Each User on Apple TV

Use account-specific storage to segregate data on a multiuser system.

Supporting Multiple Users in Your tvOS App

Store separate data for each user with the new Runs as Current User capability.

class TVUserManager

An object that indicates how to store preferences for multiple people on a shared device.