Sample Code

Updating your app package installer to use the new Service Management API

Learn about the Service Management API with a GUI-less agent app.

Download

macOS 13.0+ | Xcode 15.0+

Overview

This sample code project demonstrates how to structure your app package installer and use the new Service Management APIs. The sample project places the helper executable's `launchd` property lists in the helper executable's bundle, rather than relying on shared locations like `/Library/LaunchDaemons`. The advantages of this approach include containing your `LaunchDaemon` and `LaunchAgent` property lists in a fully codesigned app bundle that neither the system nor a third party can modify without breaking the code signature. Users can also see which app is providing the launch daemons or launch agents by choosing System Settings > General > Login Items.

This sample project contains three targets:

- `SampleLaunchAgent`, which builds a sample `LaunchAgent` helper executable.
- `SMAppServiceSampleCode`, which builds a GUI-less app that contains the sample launch agent binary property list and an executable to register the agent at install time.
- `SMAppServiceSamplePackage`, which builds the `SMAppServiceSample.pkg` installer you use to install the `SampleLaunchAgent` helper executable.

In addition to demonstrating the new package structure, this sample demonstrates the APIs for registering helper executables. To register a launch agent, the sample creates an agent object, `SMAppService.agent(plistName:)` and then calls `register()`.

```
let service = SMAppService.agent(plistName: "com.xpc.example.agent.plist")

do {
    try service.register()
    print("Successfully registered \(service)")
} catch {
    print("Unable to register \(error)")
    exit(1)
}
```

To unregister a launch agent, the sample creates an agent object, `SMAppService.agent(plistName:)` and then calls `unregister()`.

```
let service = SMAppService.agent(plistName: "com.xpc.example.agent.plist")

do {
    try service.unregister()
    print("Successfully unregistered \(service)")
} catch {
    print("Unable to unregister \(error)")
    exit(1)
}
```

To determine the authorization state of a launch agent, the sample creates an agent object, `SMAppService.agent(plistName:)` and then calls `status()` to determine the helper executable's authorization state.

```
let service = SMAppService.agent(plistName: "com.xpc.example.agent.plist")

print("\(service) has status \(service.status)")
```

Configure the sample code project

To set the Team ID for the sample app and package targets, follow these steps:

1. Open `SMAppServiceSampleCode.xcodeproj` in Xcode.
2. Select the `SMAppServiceSampleCode` target from the project editor.
3. Click Signing & Capabilities.
4. Choose your team from the Team pop-up menu.

5. Select the SMAppServiceSamplePackage from the project editor and repeat step 4.

Build the sample app service

1. Open Terminal and change to the directory into which you downloaded the UpdatingYourApp PackageInstallerToUseTheNewServiceManagementAPI source code.
2. Run the command `xcodebuild -target SMAppServiceSamplePackage`.

Xcode builds the SMAppServiceSamplePackage installer package, and the process concludes with Xcode printing `** BUILD SUCCEEDED **` to the terminal.

> **Important**
>
> When building your own projects, follow the packaging steps in the `generatePackage.zsh` script to generate a correctly configured installer package.

Install the login item and launch agent

1. In Terminal, navigate to the `build/Release` directory using the command `cd build/Release`.
2. Open the build directory in Finder with the command `open .`.
3. The Finder window contains all the products that the Xcode build process creates, including a package installer named SMAppServiceSample. Double-click the installer icon to run it, and follow the instructions to approve the installation of the SMAppServiceSampleCode login item and its supporting SampleLaunchAgent helper executable.

Run the sample launch agent

The installer adds a new login item in System Settings and installs the sample launch agent SampleLaunchAgent in the directory /Library/Application Support/YOURDEVELOPERNAME. Run the SampleLaunchAgent executable from the Terminal by invoking the command line app inside the app bundle with an additional command argument:

```
/Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode
.app/Contents/MacOS/SMAppServiceSampleCode COMMAND
```

The SMAppServiceSampleCode app supports the following commands:

- `register` — Registers the SampleLaunchAgent.
- `unregister` — Unregisters the SampleLaunchAgent.

- `status` — Checks the authorization status of the `SampleLaunchAgent`.
- `test` — Sends an XPC message to the `SampleLaunchAgent` and displays the reply.

The output from the app resembles the following, for each of the app's commands:

```
% /Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode.app/Contents
Successfully registered LaunchAgent(com.xpc.example.agent.plist)

% /Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode.app/Contents
LaunchAgent(com.xpc.example.agent.plist) has status SMAppServiceStatus(rawValue: 1)

% /Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode.app/Contents
Received "Hello World"

% /Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode.app/Contents
Successfully unregistered LaunchAgent(com.xpc.example.agent.plist)
```

Running the app before registering the service, or after unregistering the service, results in an error.

```
% /Library/Application Support/YOURDEVELOPERNAME/SMAppServiceSampleCode.app/Contents
Unable to create xpc_session <OS_xpc_rich_error: Can Retry: 0. Description:"Underlyi
```

Uninstall the login item and launch agent

To uninstall the sample app's login item from System Settings, as well as the `SampleLaunchAgent`, use the following command:

```
sudo rm -rf /Library/Application Support/YOURDEVELOPERNAME
```

The login item may remain visible in the System Settings > General > Login Items for some time. This is expected because the system removes deleted items as part of its maintenance processes overnight. To remove all third-party login items and reset the System Settings > General > Login Items to its installation defaults, use the `sfltool`: `sudo sfltool resetbtm`.

See Also

Essentials

Updating helper executables from earlier versions of macOS

Simplify your app's helper executables and support new authorization controls.