Toggle navigation Sparkle

- Home
- About
- Documentation
- GitHub Project

Basic Setup

If your app already has an older version of Sparkle, see upgrading from previous versions.

Note that Sparkle does not yet support sandboxed applications.

1. Add the Sparkle framework to your project

If you use CocoaPods:

- Add `pod 'Sparkle'` to your Podfile.
- Add or uncomment `use_frameworks!` in your Podfile.
- Xcode 7 and older require more steps.

If you don't have CocoaPods, then add Sparkle manually:

- Get the latest version of Sparkle.
- Link the Sparkle framework to your app target:
  - Drag `Sparkle.framework` into the `Frameworks` folder of your Xcode project.
  - Be sure to check the "Copy items into the destination group's folder" box in the sheet that appears.
  - Make sure the box is checked for your app's target in the sheet's `Add to targets` list.
- Make sure the framework is copied into your app bundle:
  - Click on your project in the Project Navigator.
  - Click your target in the project editor.
  - Click on the `Build Phases` tab.
  - Choose `Editor › Add Build Phase › Add Copy Files Build Phase`.
  - Click the disclosure triangle next to the new build phase.
  - Choose `Frameworks` from the Destination list.
  - Drag Sparkle.framework from the Project Navigator left sidebar to the list in the new `Copy Files` phase.
- In `Build Settings` tab set "`Runpath Search Paths`" to `@loader_path/../Frameworks` (for non-Xcode projects add the flags `-Wl,-rpath,@loader_path/../Frameworks`).
- If you have your own process for copying/packaging your app make sure it preserves symlinks!

2. Set up a Sparkle updater object

These instructions are for regular .app bundles. If you want to update a non-app bundle, such as a Preference Pane or a plug-in, follow step 2 for non-app bundles.

- Open up your MainMenu.xib.
- Choose `View › Utilities › Object Library...`

- Type “Object” in the search field under the object library (at the bottom of the right sidebar) and drag an Object into the left sidebar of the document editor.
- Select the Object that was just added.
- Choose `View › Utilities › Identity Inspector`.
- Type `SUUpdater` in the `Class` box of the `Custom Class` section in the inspector.
- If you’d like, make a “`Check for Updates...`” menu item in the application menu; set its target to the `SUUpdater` instance and its action to `checkForUpdates:`.

3. Segue for security concerns

Since Sparkle is downloading executable code to your users’ systems, you must be very careful about security. To let Sparkle know that a downloaded update is not corrupted and came from you (instead of a malicious attacker), we recommend:

- Serve updates over HTTPS.
  - Your app *will not update on macOS 10.11* unless you comply with Apple’s App Transport Security requirements. HTTP requests will be rejected by the system unless an exception is added within your app. App Transport Security has other specific requirements too, so please test updating your app on 10.11 even if you already are serving over HTTPS!
  - You can get free certificates from Let’s Encrypt, and test server configuration with ssltest.
- Sign the application via Apple’s Developer ID program.
- Sign the published update archive with Sparkle’s DSA signature.
  - Updates using Installer package (`.pkg`) *must* be signed with DSA.
  - Binary Delta updates *must* be signed with DSA.
  - Updates of preference panes and plugins *must* be signed with DSA.
  - DSA signatures are optional for updates using regular app bundles that are signed with Apple code signing (Apple’s Developer ID program or your own certificate), but we still recommended DSA signatures as a backup.

DSA signatures

To prepare signing with DSA signatures:

- First, make yourself a pair of DSA keys. This needs to be done only once. Sparkle includes a tool to help: (from the Sparkle distribution root):
  `./bin/generate_keys`
- Back up your private key (dsa_priv.pem) and **keep it safe.** You don’t want anyone else getting it, and if you lose it, you may not be able to issue any new updates.
- Add your public key (dsa_pub.pem) to the `Resources` folder of your Xcode project.
- Add an `SUPublicDSAKeyFile` key to your `Info.plist`; set its value to your public key’s filename—unless you renamed it, this will be `dsa_pub.pem`.

Apple code signing

If you are code-signing your application via Apple’s Developer ID program, Sparkle will ensure the new version’s author matches the old version’s. Sparkle also performs basic (but not deep) validation for testing if the new application is archived/distributed correctly as you intended.

- Note that embedding the Sparkle.framework into the bundle of a Developer ID application requires that you code-sign the framework with your Developer ID keys. Xcode should do this automatically if you let it “`Code Sign on Copy`” Sparkle’s framework.
- You can diagnose code signing problems with RB App Checker app and by checking logs in the Console.app.

If you both code-sign your application and include a public DSA key for signing your update archive, Sparkle allows issuing a new update that changes either your code signing certificate or your DSA keys. Note however this is a last resort and should *only* be done if you lose access to one of them.

4. Distributing your App

For best compatibility with macOS Sierra and later you should use *signed* disk image (DMG) for distribution of apps from your product's website. The system will quarantine (translocate) apps downloaded from the Internet, unless they're from a signed DMG, installer package, or are moved using Finder.

If you distribute your app as a disk image (DMG):

- Add an `/Applications` symlink in your DMG, or otherwise encourage the user to copy the app out of it (the app can't be updated when it's launched straight from the disk).
- Make sure the DMG is signed with a Developer ID and use macOS 10.11.5 or later to sign it (an older OS may not sign correctly). Signed DMG archives are backwards compatible.
- If you do not sign the DMG, avoid placing your app inside another folder in your archive.

If you distribute your app as a ZIP or a tar archive:

- Encourage users to move the app to `/Applications` (e.g. use LetsMove). The system will not allow the app to update itself until it's moved.
- Avoid placing your app inside another folder in your archive, because copying of the folder as a whole doesn't remove the quarantine.
- Avoid putting more than just the single app in the archive.

Sparkle supports updating from DMG, ZIP archives, tarballs, and installer packages, so you can generally reuse the same archive for distribution of your app on your website as well as Sparkle updates.

5. Publish your appcast

Sparkle uses appcasts to get information about software updates. An appcast is an RSS feed with some extra information for Sparkle's purposes.

- Add a `SUFeedURL` key to your `Info.plist`; set its value to a URL where your appcast will be hosted, e.g. `https://yourcompany.example.com/appcast.xml`. We strongly encourage you to use HTTPS URLs for the appcast.
- Remember that your bundle must have a properly formatted `CFBundleVersion` key in your `Info.plist`.

If you update regular app bundles and you have set up DSA signatures, you can use a tool to generate appcasts automatically:

- Build your app and compress it (e.g. in a DMG/ZIP/tar.bz2 archive), and put the archive in a new folder. This folder will be used to store all your future updates.

- Run `generate_appcast` tool from Sparkle's distribution archive specifying the path to your DSA private key, and the folder with update archives:

  ```
  ./bin/generate_appcast /path/to/your/dsa_priv.pem /path/to/your/updates_folder/
  ```

- The tool will generate the appcast file (using filename from `SUFeedURL`) and also `*.delta` update files. Upload your archives, the delta updates and the appcast to your server.

You can also create the appcast file manually:

- Make a copy of the sample appcast included in the Sparkle distribution.
- Read the sample appcast to familiarize yourself with the format, then edit out all the items and add one for the new version of your app by following the instructions at Publishing an update.

6. Test Sparkle out

- Use an older verison of your app, or if you don't have one yet, make one by editing `Info.plist` and change `CFBundleVersion` to a lower version.
    - A genuine older version of the app is required to test delta updates, because Sparkle will ignore the delta update if the app doesn't match update's checksum.
    - Editing `CFBundleVersion` of the latest version of the app is useful for testing the latest version of Sparkle framework.
- Run the app, then quit. Sparkle doesn't ask the user about updates until the *second* launch, in order to make your users' first-launch impression cleaner.
- Run the app again. The update process should proceed as expected.

Update process will be logged to `Console.app`. If anything goes wrong, you should find detailed explanation in the log.

Next steps

That's it! You're done! You don't have to do any more. But you might want to:

- Adjust Sparkle's settings and behavior for your product.
- Add update settings to your preferences panel.
- Add binary delta updates to your application.
- Learn about gathering anonymous statistics about your users' systems.

This website is open source.