Startup Items

Deprecation Note: Startup items are a deprecated technology. Launching of daemons through this process may be removed or eliminated in a future release of OS X.

Unless your software requires compatibility with OS X v10.3 or earlier, use the `launchd` facility instead. For more information, see Creating Launch Daemons and Agents.

A startup item is a specialized bundle whose code is executed during the final phase of the boot process, and at other predetermined times (see Managing Startup Items). The startup item typically contains a shell script or other executable file along with configuration information used by the system to determine the execution order for all startup items.

The `/System/Library/StartupItems` directory is reserved for startup items that ship with OS X. All other startup items should be placed in the `/Library/StartupItems` directory. Note that this directory does not exist by default and may need to be created during installation of the startup item.

Anatomy of a Startup Item

Unlike many other bundled structures, a startup item does not appear as an opaque file in the Finder. A startup item is a directory whose executable and configuration property list reside in the top-level directory. The name of the startup item executable must match the name of the startup item itself. The name of the configuration property list is always `StartupParameters.plist`. Depending on your needs, you may also include other files in your startup item bundle directory.

```
MyStartupItem/
    |
    |- MyStartupItem
    \- StartupParameters.plist
```

To create your startup item:

1. Create the startup item directory. The directory name should correspond to the behavior you're providing.

   Example: `MyDBServer`

2. Add your executable to the directory. The name of your executable should be exactly the same as the directory name. For more information, see Creating the Startup Item Executable.
3. Create a property list with the name `StartupParameters.plist` and add it to the directory. For information on the keys to include in this property list, see Specifying the Startup Item Properties .

   Example: `MyDBServer/StartupParameters.plist`

4. Create an installer to place your startup item in the `/Library/StartupItems` directory of the target system. (Your installer may need to create this directory first.)

   Your installer script should set the permissions of the startup item directory to prevent non-root users from modifying the startup item or its contents. For more information, see Startup Item Permissions.

Important: A startup item is *not* an application. You cannot display windows or do anything else that you can't do while logged in via ssh. If you want to launch an application after login, you should install a login item instead. For more information, see Adding Login Items.

Creating the Startup Item Executable

The startup item executable can be a binary executable file or an executable shell script. Shell scripts are more commonly used because they are easier to create and modify.

If you are implementing your startup item executable as a shell script, OS X provides some code to simplify the process of creating your script. The file `/etc/rc.common` defines routines for processing command-line arguments and for gathering system settings. In your shell script, source the `rc.common` file and call the `RunService` routine, passing it the first command-line argument, as shown in the following example:

```
#!/bin/sh
. /etc/rc.common


# The start subroutine
StartService() {
    # Insert your start command below.  For example:
```

```
# The stop subroutine
StopService() {
    # Insert your stop command(s) below.  For example:
    killall -TERM mydaemon
    sleep 10
    killall -9 mydaemon
    # End example.
}

# The restart subroutine
RestartService() {
    # Insert your start command below.  For example:
    killall -HUP mydaemon
    # End example.
}

RunService "$1"
```

The `RunService` routine looks for `StartService`, `StopService`, and `RestartService` functions in your shell script and calls them to start, stop, or restart your services as needed. You must provide implementations for all three routines, although the implementations can be empty for routines whose commands your service does not support.

If your startup-item executable contains code that might take a long time to finish, consider spawning off a background process to run that code. Performing lengthy startup tasks directly from your scripts delays system startup. Your startup item script should execute as quickly as possible and then exit.

For more information about writing shell scripts, see *Shell Scripting Primer*.

Note: Most Apple-provided startup items have a test in the script to check to see if a particular variable is set to prevent automatic starting of daemons unless they are enabled (usually in the System Preference sharing pane).

To enable or disable a daemon that does not have a GUI checkbox, you must add or modify these variables directly by editing the file `/etc/hostconfig`. Note that this file is writable only by the `root` user, so this technique is discouraged. (Use a `launchd` daemon instead, if at all possible.)

Specifying the Startup Item Properties

The configuration property list of a startup item provides descriptive information about the startup item, lists the services it provides, and lists its dependencies on other services. OS X uses the service and dependency information to determine the launch order for startup items. This property list is stored in ASCII format (as opposed to XML) and can be created using the Property List Editor application.

Note: In OS X v10.4 and later, the dependency information for many stubbed-out system startup items is still present in case other startup items depend on it.

Table A-1 lists the key-value pairs you can include in your startup item's `StartupParameters.plist` file. Each of the listed arrays contains string values. You can use the Property List Editor application that comes with the Xcode Tools to create this property list. When saving your property-list file, be sure to save it as an ASCII property-list file.

Table A-1 StartupParameters.plist key-value pairs

Key	Type	Value
Description	String	A short description of the startup item, used by administrative tools.
Provides	Array	The names of the services provided by this startup item. Although a startup item can potentially provide multiple services, it is recommended that you limit your startup items to only one service each.
Requires	Array	The services provided by other startup items that must be running before this startup item can be started. If the required services are not available, this startup item is not run.
Uses	Array	The services provided by other startup items that should be started before this startup item, but which are not required. The startup item should be able to start without the availability of these services.

For example, here is an old-style plist:

```
  Requires        = ("Network");
  Uses            = ("Network");
  OrderPreference = "Late";
  Messages =
  {
    start = "Starting Software Update service";
    stop  = "Stopping Software Update service";
  };
}
```

And here is an XML plist example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs/PropertyList.dtd">
<plist version="0.9">
    <dict>
        <key>Description</key>
        <string>Apple Serial Terminal Support</string>
        <key>OrderPreference</key>
        <string>Late</string>
        <key>Provides</key>
        <array>
                <string>Serial Terminal Support</string>
        </array>
        <key>Uses</key>
        <array>
                <string>SystemLog</string>
        </array>
    </dict>
</plist>
```

The service names you specify in the `Requires` and `Uses` arrays may not always correspond directly to the name of the startup item that provides that service. The `Provides` property specifies the actual name of the service provided by a startup item, and while this name usually matches the name of the startup item, it is not required to do so. For example, if the startup item launches multiple services, only one of those services can have the same name as the startup item.

If two startup items provide a service with the same name, the system runs only the first startup item it finds with that name. This is one of the reasons why your own startup items should launch only one service. If the name of only one of the services matches the name of another service, the entire startup item might not be executed and neither service would be launched.

The values of the `Requires` and `Uses` keys do not guarantee a particular launch order.

In OS X v10.4 and later, most low-level services are started with `launchd`. By the time your startup item starts executing, `launchd` is running, and any attempt to access any of the services provided by a `launchd` daemon will result in that daemon starting. Thus, you can safely assume (or at least pretend) that any of these services are running by the time your startup item is called.

For this reason, with few exceptions, the `Requires` and `Uses` keys are largely irrelevant after OS X v10.3 except to support service dependencies between two or more third-party startup items.

Managing Startup Items

During the boot process, the system launches the available startup items, passing a `start` argument to the startup item executable. After the boot process, the system may run the startup item executable again, this time passing it a `restart` or `stop` argument. Your startup item executable should check the supplied argument and act accordingly to start, restart, or stop the corresponding services.

Note: In general, with the exception of daemons provided with OS X, the system will only run your startup script with `start` or `stop` arguments (at boot and shutdown, respectively). Users, however, may elect to use the `restart` argument.

You should not make any assumptions about the order in which daemons will be shut down.

If you want to start, restart, or stop startup items from your own scripts, you can do so using the `SystemStarter` program. To use `SystemStarter`, you must execute it with two parameters: the desired action and the name of the service provided by the startup item. For example, to restart the Apache Web server (prior to OS X v10.4), you would execute the following command:

Startup items should always respect the arguments passed in by `SystemStarter`. However, the response to those arguments is dependent on the startup item. The stop and restart options may not make sense in all cases. Your startup item could also support the restart option using the existing code for stopping and starting its service.

Displaying and Localizing a Startup Message

When your startup item starts at boot time, you may (if desired) display a message to the user. To do this, use the `ConsoleMessage` command. (You can use this command even if the computer is not starting up, but the user will not see it unless the Console application is running.)

For example:

```
ConsoleMessage "MyDaemon is running.  Better go catch it."
```

If you want to localize the message displayed when a startup item starts, you must create a series of property list files with localized versions of the strings. Each of these files must be named `Localizable.strings`, and must be in a localized project directory whose name is based on the name of a language or locale code for the desired language. These folders, in turn, must be in a folder called `Resources` in the startup item folder.

For example, you might create a tree structure that looks like this:

```
MyDaemon
    |
    |- MyDaemon
    |- StartupParameters.plist
    \- Resources
            |
            |- English.lproj
            |- French.lproj
            |- German.lproj
            \- zh_CN.lproj
```

Within each of these localizable strings files, you must include a dictionary whose keys map an English string to a string in another language. For example, the French version of the `PrintingServices` localization file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
    <dict>
        <key>Starting printing services</key>
        <string>Démarrage des services d’impression</string>
    </dict>
</plist>
```

Whenever the `ConsoleMessage` command is passed the string “Starting printing services”, if the user’s language preference is French, the user will instead see “Démarrage des services d’impression” at startup. C’est très bien!

The value of the *key* field must precisely match the English string printed by your startup item using `ConsoleMessage`.

See the manual page for `locale` for more information about locale codes.

Startup Item Permissions

Because startup items run with root authority, you must make sure your startup item directory permissions are set correctly. For security reasons, your startup item directory should be owned by root, the group should be set to wheel, and the permissions for the directory should be 755 (rwxr-xr-x). This means that only the root user can modify the directory contents; other users can examine the directory and view its contents, but cannot modify them. The files inside the directory should have similar permissions and ownership. Thus, the file listing for the Apache startup item directory is as follows:

```
./Apache:
total 16
drwxr-xr-x    4 root  wheel   136 Feb 14 14:33 .
drwxr-xr-x   21 root  wheel   714 Feb 14 15:03 ..
-rwxr-xr-x    1 root  wheel  1253 Feb 10 19:31 Apache
```

your software. To avoid this, be sure to set the permissions during installation.

Previous Next