

Google Chrome Install Recipe (macOS)

Description

Installation of Chrome or any app that simply goes in /Applications can be fairly simple to do with a Fileset, but if the application is presented as a PKG that does other things besides simply copying a file to /Applications, then this can be a good example of how to accomplish the install. Note that this Fileset also will download the latest Google Chrome at install time, so depending on your use case, this may not be a good solution. An alternative would be to put the PKG inside of the FileSet and use the same script to install it or make a PKG-based FileSet in FileWave. There are many roads to success. This example will show several features of Filesets for teaching purposes.

Ingredients

- FW Central
- Chrome already as a Fileset (below)



Directions

1. Download the Chrome Fileset zip file, expand and drag the Fileset into FileWave > Filesets
2. Create an Association between the Fileset and one or more macOS systems
3. See Chrome installed on them.

What does this Fileset do?

The Fileset is entirely 2 scripts. If you highlight the Fileset in the Native Admin and pick the Scripts button in the toolbar you can edit the 2 scripts. The first is the Activation Script. This will create a directory in /private/tmp/ and download Chrome to it. It will install and then delete the installer from that folder. Because of the way this works you may want to make this script a little fancier with error handling, but the below works and is a good starting point. You will also see that it logs what it is doing to /Library/Libgs/GoogleChromeInstallScript.log so that you can review the date and time that events happened.

```
#!/bin/zsh

pkgfile="GoogleChrome.pkg"
logfile="/Library/Logs/GoogleChromeInstallScript.log"
url='https://dl.google.com/chrome/mac/stable/gcem/GoogleChrome.pkg'

/bin/echo "--" >> ${logfile}
/bin/echo "`date`: Downloading latest version." >> ${logfile}
mkdir /private/tmp/chrome_install/
/usr/bin/curl -s -o /private/tmp/chrome_install/${pkgfile} ${url}

/bin/echo "`date`: Installing..." >> ${logfile}
/usr/sbin/installer -pkg /private/tmp/chrome_install/GoogleChrome.pkg -target /

/bin/sleep 5

/bin/echo "`date`: Deleting package installer." >> ${logfile}
/bin/rm -rf /private/tmp/chrome_install

exit 0
```

Note that in these days of Intel vs. Apple Silicon you may want to add something to behave differently depending on which processor is present below is an easy way in scripting to do this;

```
if [[ $(uname -p) == 'arm' ]]; then
    echo M1
else
    echo Intel
fi
```

The second script is for when you remove the Association for the Fileset. It will kill Chrome if it is running and then delete it from /Applications. If you don't want this behavior then just remove that script from the Fileset.

```
#!/bin/zsh

logfile="/Library/Logs/GoogleChromeInstallScript.log"

/bin/echo "--" >> ${logfile}
/bin/echo "`date`: Removing Google Chrome." >> ${logfile}

killall Google\ Chrome

rm -rf "/Applications/Google Chrome.app"

exit 0
```

For Requirements, I set it for macOS and only the most recent versions since (1) Anything below macOS 10.14.x is insecure to use, and (2) I don't know that Chrome will work well on older macOS, but adjust this to your needs.

Requirements

☒ Platform

☐ Windows

☒ macOS

☐ Android

☐ Architecture

☒ Intel ☒ 64 bit

☒ Apple Silicon ☒ 32 bit

☐ Min. Memory

0 MB

☒ System Version

macOS	Windows	Windows Server	Android	
<input type="checkbox"/> 10.7.x	<input type="checkbox"/> XP	<input type="checkbox"/> 2008 R2	<input type="checkbox"/> 4.1	<input checked="" type="checkbox"/> Use/install on newer OS versions
<input type="checkbox"/> 10.8.x	<input type="checkbox"/> Vista	<input type="checkbox"/> 2012	<input type="checkbox"/> 4.2	
<input type="checkbox"/> 10.9.x	<input type="checkbox"/> 7	<input type="checkbox"/> 2012 R2	<input type="checkbox"/> 4.3	
<input type="checkbox"/> 10.10.x	<input type="checkbox"/> 8	<input type="checkbox"/> 2016	<input type="checkbox"/> 4.4	
<input type="checkbox"/> 10.11.x	<input type="checkbox"/> 8.1	<input type="checkbox"/> 2019	<input type="checkbox"/> 5.0	
<input type="checkbox"/> 10.12.x	<input type="checkbox"/> 10.0	<input type="checkbox"/> 2022	<input type="checkbox"/> 5.1	
<input type="checkbox"/> 10.13.x	<input type="checkbox"/> 11.0		<input type="checkbox"/> 6.0	
<input checked="" type="checkbox"/> 10.14.x			<input type="checkbox"/> 7.0	
<input checked="" type="checkbox"/> 10.15.x			<input type="checkbox"/> 7.1	
<input checked="" type="checkbox"/> 11.x				
<input checked="" type="checkbox"/> 12.x				

☐ Evaluate requirements on change and uninstall active Fileset if they failed

For Delete Files, I have it purging the download files so that when the Fileset activates, the folder is clear every time.

Delete Files

☒ Delete the paths below upon Fileset activation

☐ Execute at every verification

/private/tmp/chrome_install/

On the Kiosk tab, I set this to Applications and put a description in. Using a Kiosk Association is a great way to make an application installable by non-admins and they can even uninstall it if they aren't using it any more.

Properties


Requirements

Dependencies


Delete Files

Kiosk

Kiosk Icon



☐ Staff Rating



Category

[\(edit categories\)](#)

Applications

Utilities

Scripts

Policies

Profiles

OS Updates

Media

Title:

Google Chrome (macOS)

Description:

B

I

U

This will install Google Chrome on your device. If you pick to Uninstall from the Kiosk it will remove Chrome from your system.

So with this example, you can see how you might create a scripted install of an application and handle uninstall of it as well. Keep in mind you don't have to do this exactly as I did, and these concepts can be applied to other applications.

After installing Chrome, some configuration may be desirable, examples may include:

* Block auto updates

* Set home page

* Disable Welcome Page

This article discusses configuration options: [Google Chrome Configuration Recipe](#)

Related Content

- [Google Chrome Configuration recipe \(macOS\)](#)
- [Google Chrome Extension Management \(macos\)](#)