

The MBS Xojo Updater Kit

Version 2.1, © 2010-2017 by Christian Schmitz

About the Updater Kit	3
Parts of the Updater Kit	4
<i>AppCast Creator</i>	4
<i>Drop File to See MD5 Checksum.rbp</i>	4
<i>Manual</i>	4
<i>New Version App.rbp</i>	4
<i>On the Server</i>	4
<i>Sparkle</i>	5
<i>Updater Example.rbp</i>	5
<i>UpdaterEngine</i>	5
<i>Windows Installer</i>	5
Get the sample application running	6
Create your own	8
<i>AppCast.xml</i>	10
Build Step	12
Our Build Script	15
32-bit/64-bit transition	17
Changes	18
<i>Version 1.0</i>	18
<i>Changes for Version 1.1</i>	18
<i>Changes for Version 1.2</i>	18
<i>Changes for Version 1.3</i>	18
<i>Changes for Version 1.4</i>	18

<i>Changes for Version 1.5</i>	19
<i>Changes for Version 1.6</i>	19
<i>Changes for Version 2.0</i>	19
<i>Changes for Version 2.1</i>	19
Troubleshooting	20
<i>No update is found</i>	20
<i>Update verification fails</i>	20
Requirements	21
License	22

About the Updater Kit

There are a few things each application needs and one is an automatic update system. One that you can easily add to all of your applications.

Updater Kit is based on the Sparkle Framework on Mac. But as cross platform development is important, we add our own Windows part. For Windows the best way is to simply download an installer for the updates. The installer can then handle all the permissions things like asking for administration rights to actually install something.

Our Updater Kit can be integrated into Xojo easily. And with our Build Step to add necessary resources to the compiled application, you can automate the build process. You can integrate the AppCast Creator into your packaging and upload code and generate the xml files on the fly with all the signature/checksum stuff.

Linux may be added in the future. Email us at support@monkeybreadsoftware.de if you would require a Linux version of the Update Kit.

Parts of the Updater Kit

Our Updater Kit contains this files and folders:

AppCast Creator

This project will help create AppCast files for you. You may want to extend this project and add it to your release cycle. The digital signature and the MD5 checksum are automated in the Update Kit.

Build Step / PatchMacApp

This is an application to add the resources needed for the update engine to a Mac application. This includes the Sparkle Framework, the public key and the info.plist entries.

We have three variations of our helper application. First we have the PatchMacApp in a GUI and a console variation for you. Second we have the Build Step folder with a console application we use for a build step in Xojo build automation. We include a script for you and a test project.

You will certainly want to customize this projects for your needs. You should change the configuration options in the build script for the build automation or the pathes and the URL in the PatchMacApp project.

Drop File to See MD5 Checksum.rbp

A simple tool to show the MD5 Checksum of a given file. If you use the AppCast Creator, you won't need it, but if you run AppCast manually you should also calculate the md5 checksum. Of course you can do the same in the terminal on Mac OS X with the md5 command. Type *"md5"* a space and the path to the file. For the path you can drop the file on the terminal window.

Manual

This PDF file is the manual for the Updater Kit which you are reading now.

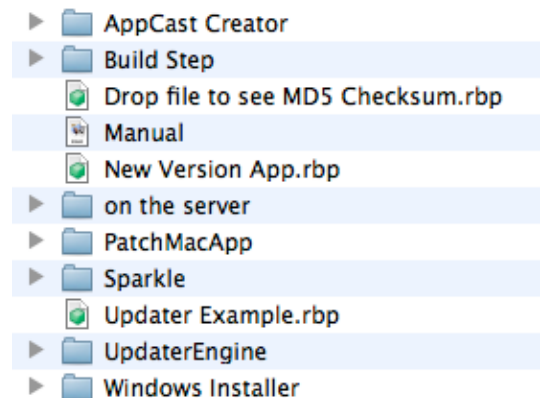
New Version App.rbp

This is our test application project for the updating. We compiled this project and put it on the server as the update to download. So once you updated our test project, you get this application running.

On the Server

This folder contains the files located on the monkeybreadsoftware.de server in the / UpdaterTest folder.

In this folder there is a zip file called UpdaterExampleMac.zip, which is the zip archive with the new version of the example application. The file UpdaterExampleWin.exe is an installer to install the new version of the example application for Windows. The AppCast.xml is the xml file with the updates readable for the Update Kit. The two description files on the server are for the German and English description which is displayed in the updater window.

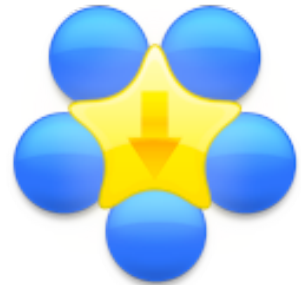


You can take a look: <http://www.monkeybreadsoftware.de/UpdaterTest/>

Sparkle

This folder contains files from the Sparkle updater engine. You should use the ruby scripts there to create your own pair of keys for the signing process.

To create a private key, you need to open the terminal application on Mac. Type "cd" for the change directory command. Drop your Sparkle folder on the terminal window and press "Return." You should now be in the Sparkle folder. Next type "ruby ./generate_keys.rb" and press "Return." Two files will be generated: dsa_priv.pem and dsa_pub.pem. The dsa_priv.pem file contains your private key. Keep it secret. The other file dsa_pub.pem contains your private key which you bundle in your application.



To sign an update you run the other ruby script: sign_update.rb. So launch the terminal again and type "ruby". Now drop 3 files in the terminal window in the right order. First the sign_update.rb script file, second the application zip archive and third the dsa_priv.pem file with your private key. If you drop them right, the terminal command line is complete with all three paths and you can press return. A typical signature looks like "MC4CFQDPWH6GZIIIRS9Icbbk+Q5XrddKphAIVAMWgyIFuB11TKzDMm5mKoG6o4tWk". Basically a Base 64 encoded string. Due the way signatures work the string is different on each time you calculate it.

You can find more information here: <http://sparkle.andymatuschak.org/>

Updater Example.rbp

This is the example project to show you how this updates work. You can compile it. Once you run it, you can test the update engine.

UpdaterEngine

In this folder you will find files to add to your application in order to use the update engine in your application. First, there is the central module named UpdaterEngine with all the public methods. Next, there is the UpdateHTTPSsocket which is a HTTPSsocket subclass and does the downloads. If you plan to use SSL on your website to secure your downloads, you can change this class to use SecureHTTPSsocket. The UpdateWindow now is the only window our Updater Kit uses. It shows information about updates found.

Windows Installer

This folder contains files we use to create our installers. For our sample here we used Inno Setup Compiler 5.3.4, which is a free setup creation tool. You can find it here: <http://www.jrsoftware.org/isinfo.php>. Which tool you use is your own decision. As long as it creates a self containing exe file for Windows, it will work.

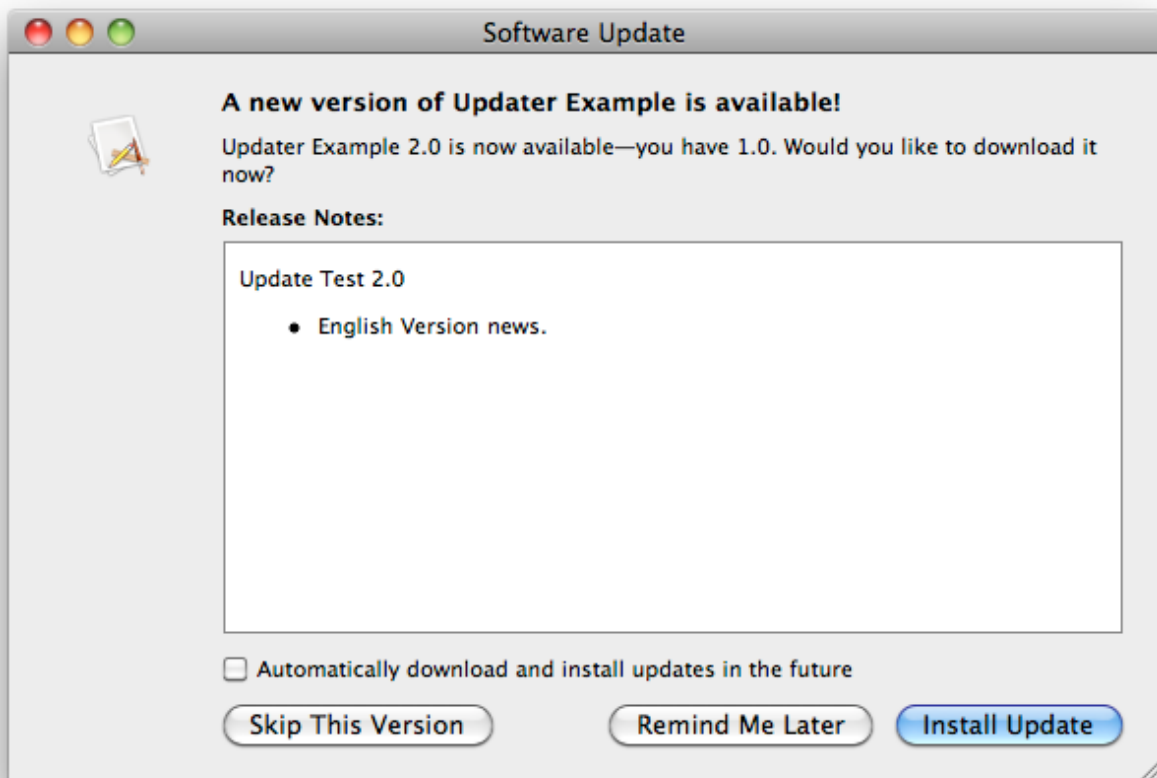
Get the sample application running

For the sample we have already a folder on the monkeybreadsoftware.de web server with the AppCast file, the archives and the description files.

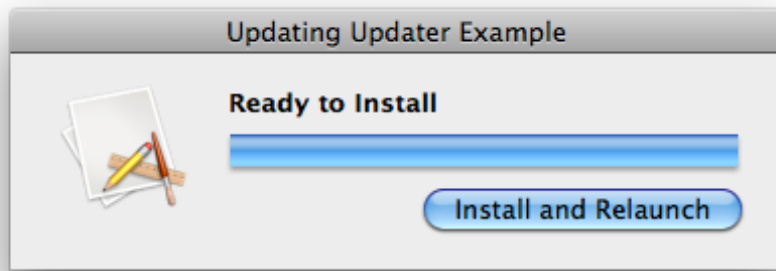
The newer versions of our test project are created using the *New Version App.rbp* project. We created a Windows installer and used the Finder compress feature on Mac to create a zip archive. Using the tool *Drop File* to see *MD5 Checksum.rbp* we got the MD5 checksum for the Windows file and using the ruby scripts in the Sparkle folder, we got a digital signature for the Mac zip archive. All those values with the correct links and size values are in the AppCast.xml file.

To try out the example, you can compile the project *Updater Example.rbp*. After you have compiled this project, you need to run the project *PatchMacApp.rbp* to install the required resources for the Sparkle engine. Once you launch the example application a second time (updates are never offered at first launch for a better use experience) you see the update window. On Mac it is the Sparkle engine and on Windows our own window.

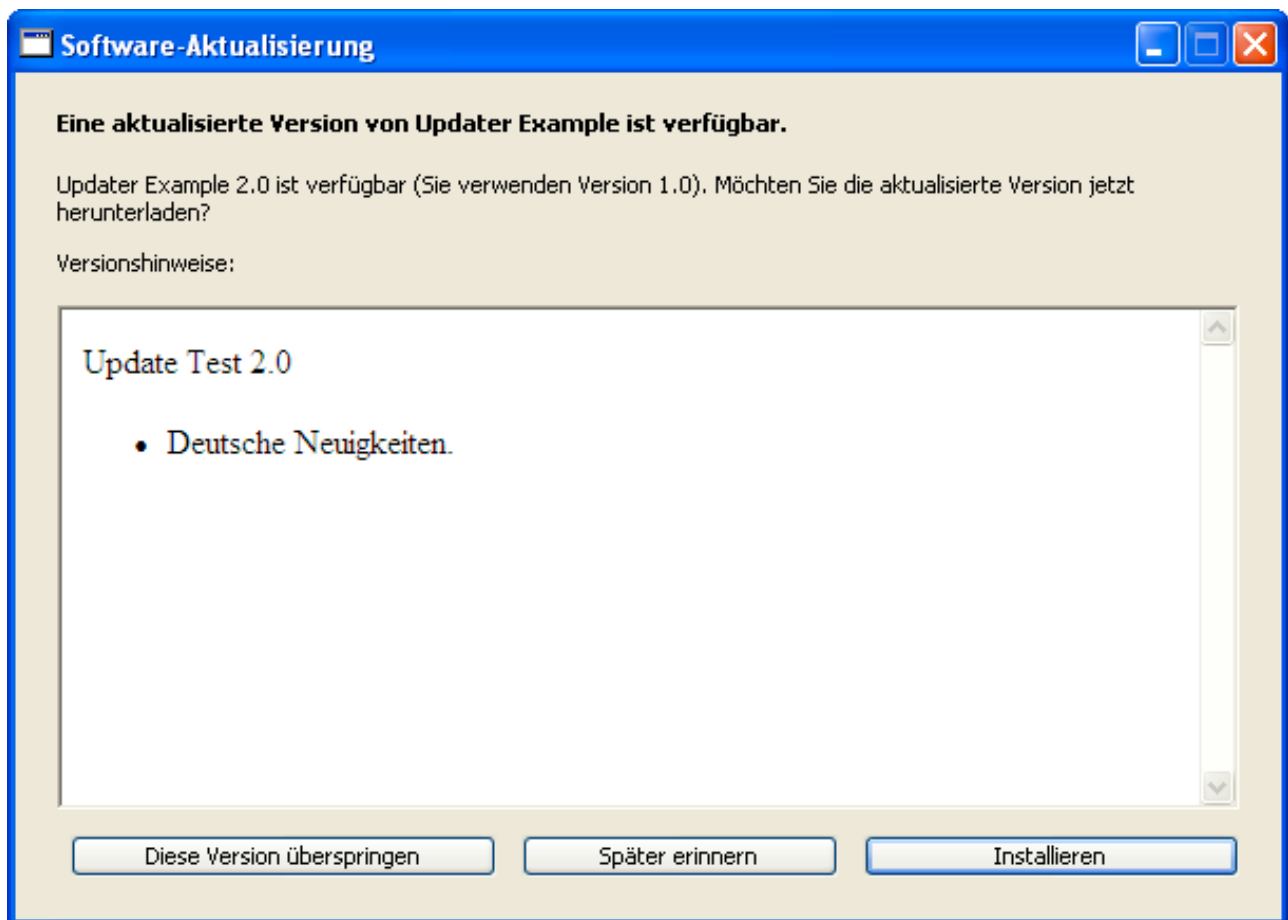
Now you can run the update.



English Dialog on Mac OS X.



English download Window on Mac OS X.



German update dialog on Windows XP.

Create your own

To get this updater running in your application you need this steps:

First you add the items from the UpdaterEngine folder to your application. This includes currently a window, a module and a class.

In your app event you add a line of code after the registration of the MBS Plugins or any other initialization stuff:

```
UpdaterEngine.Init "http://www.yourdomain.test/yourapp.xml"
```

In this line you pass the URL for your AppCast.xml file.

You can optionally pass an application display name, which is used for displaying the application name. If you don't provide a name, the Updater Kit takes the name from the application file.

```
UpdaterEngine.Init "http://www.yourdomain.test/yourapp.xml", „MyApplication“
```

Next, you may want to add some menu command or update button to your GUI and call there this line:

```
UpdaterEngine.CheckForUpdates
```

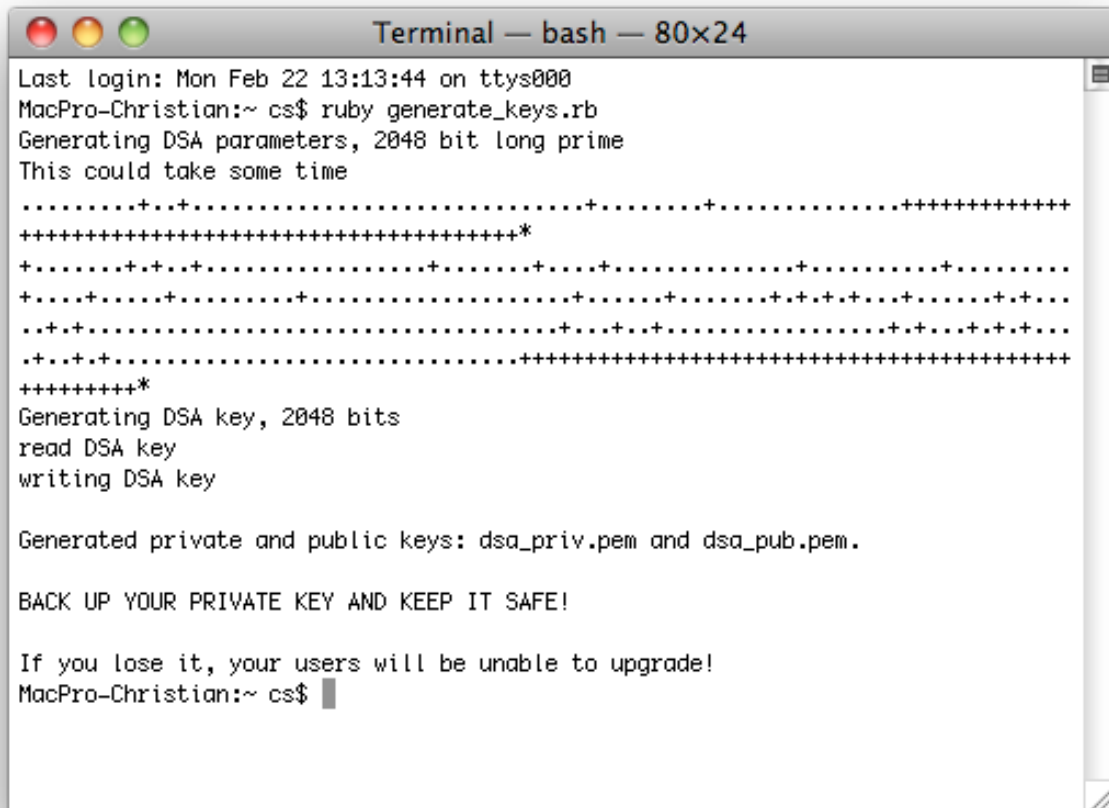
This is for manual updates showing the GUI.

Next, you want to make sure that your application has a proper version number. The version number in app.ShortVersion must be to be parsed and comparable to the newer versions you specify in your AppCast.

For Sparkle Setup you can read the following webpage: <http://sparkle.andymatuschak.org/documentation/pmwiki.php/Documentation/BasicSetup>.

The important thing to do is to create a pair of keys to digitally sign your Mac updates. So launch the terminal, and type the command `"ruby"` followed with the path to the `"generate_keys.rb"` ruby script. This will generate you the files `"dsa_priv.pem"` and `"dsa_pub.pem."`

See this screenshot:



For adding the keys, the plist URL and the Sparkle Framework to your application, you should customize the `"PatchMacApp.rb"` example project. Change the URL and the file paths so your own application is modified.

Or you use the Build Step. This way Xojó can add the keys, the plist URL and the Sparkle Framework to your application on every build.

Now you can patch your own application. Manually, with PatchMacApp or with the Build Step. After patching it should contain the Sparkle.framework in the MacOS folder, the dsa_pub.pem file in the Resources folder and the info.plist file should contain the SUFeedURL and SUPublicDSAKeyFile keys.

On Windows you create an installer for your application. For example you can use Inno Setup Compiler 5.3.4, which you can download here: <http://www.jrsoftware.org/isinfo.php>. We include a sample setup file `"Updater Example.iss"` which you can modify for your installer. Basically, you copy the `"Updater Example.iss"` file and the application folder from

your build folder to a Windows PC and let Inno Setup Compiler compress your application to a small exe file.

Next, you want to create a folder on your webspace with all the files for the update engine. There you have a zip file with the Mac application and an exe file with the Windows application installer. Also you put there a description html file for each language and the AppCast.xml file.

AppCast.xml

For your updates you need an AppCast. We have an utility called AppCast Creator which will help you in creating this xml files. It can automatically calculate the MD5 Checksum and add the digital signature for your files. If you use our AppCast Creator project, you can skip the following instructions.

The example AppCast.xml file looks like this:

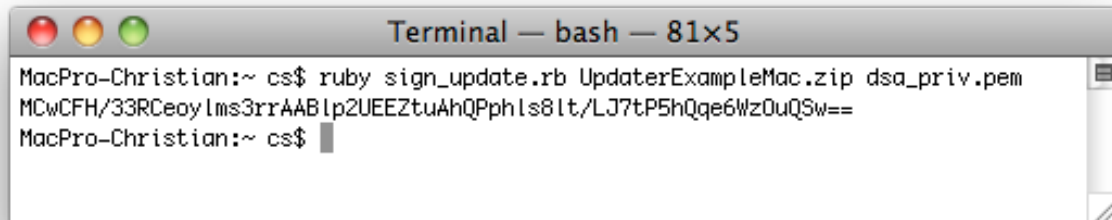
```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:sparkle="http://www.andymatuschak.org/xml-namespaces/sparkle">
  <channel>
    <title>Updates (English)</title>
    <link>http://www.yourdomain.test/yourApp/AppCast.xml</link>
    <description>Most recent changes with links to updates.</description>
    <item>
      <title>Version 2.0</title>
      <sparkle:releaseNotesLink xml:lang="en">http://www.yourdomain.test/
yourApp/description.html</sparkle:releaseNotesLink>
      <sparkle:releaseNotesLink xml:lang="de">http://www.yourdomain.test/
yourApp/description-de.html</sparkle:releaseNotesLink>
      <pubDate>21 Feb 2010 20:00:00 +0000</pubDate>
      <enclosure url="http://www.yourdomain.test/yourApp/
UpdaterExampleMac.zip"
        sparkle:version="2.0"
        sparkle:dsaSignature="MC0CFQCRwRCxIp9N+X7h54A0x1fSX/
tcdwIUf2RBJl60guIir6xotfYYipysPno="
        length="2860194"
        urlWin32="http://www.yourdomain.test/yourApp/
UpdaterExampleWin.exe"
        lengthWin32="1145333"
        MD5Win32="193941CD70FD908CC9397D09597609E1"
        type="application/octet-stream" />
    </item>
  </channel>
</rss>
```

There are channels for each product. Inside you have items, one for each version, but probably only one with the latest version. In *“sparkle:releaseNotesLink”* you give the link to a webpage for the version details which is displayed in the updater window. You can here have a *xml:lang* tag to specify the language. Next you have the *pubDate* which is the date of the publication. In the enclosure URL finally is the actual update file. There is the URL for the Mac update zip file. Note that the name of the application in the zip file must be identical to the name of the old application. The version number of this update is given here and this is the version number we compare with the *app.shortversion*. Next you find the signature for the Mac update which you get with this terminal command:

```
ruby sign_update.rb UpdaterExampleMac.zip dsa_priv.pem
```

The output is the signature as Base64 encoded string which you copy to your AppCast file. Note that you can get several signatures from this if you call it multiple times. All signatures are valid, but you need only one.

Screenshot from the Terminal:

A screenshot of a macOS Terminal window titled "Terminal — bash — 81x5". The window shows a command prompt where the user has entered the command: `MacPro-Christian:~ cs$ ruby sign_update.rb UpdaterExampleMac.zip dsa_priv.pem`. The output of the command is a long Base64 encoded string: `MCwCFH/33RCeoylms3rrAABlp2UEEZtuAhQPphls8lt/LJ7tP5hQqe6Wz0uQSw==`. The prompt then returns to `MacPro-Christian:~ cs$` with a cursor.

Next in the enclosure attributes is the `urlWin32` which is the URL to download the Windows update. This must point to an executable file. Actually the name of the file in the URL does not matter as on disc it is written in an `installer.exe` file. Next value is the `lengthWin32` field which ensures that the file is not too short. After that we have the MD5 checksum which is used to verify that the windows file contains the right content.

After you uploaded all files to your website, you can run your application in the older version and see whether it finds the update. For the upload we recommend that you upload the xml file as the last thing so no client tries to download the update while you still upload it.

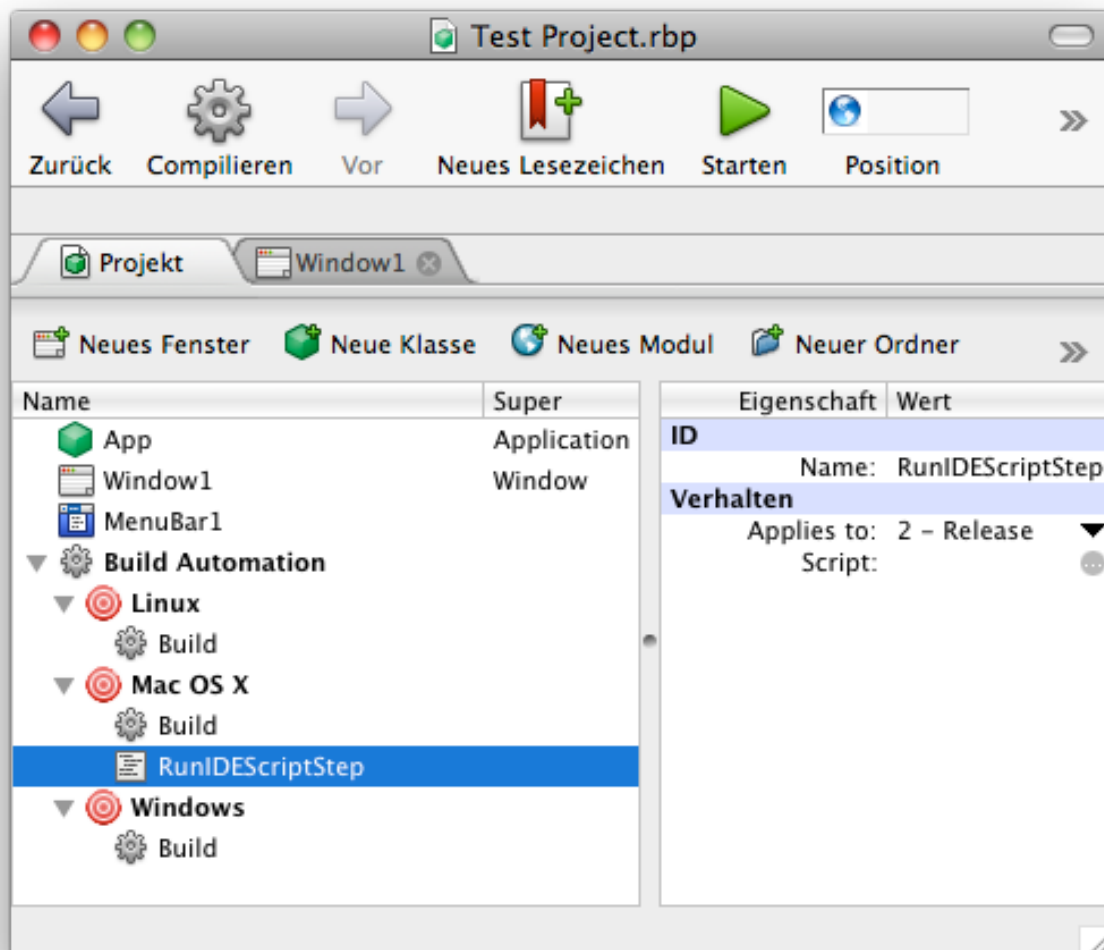
Build Step

The Build Step Script and command line tool allow you to have Xojo automatically add the Sparkle Framework, the public key and the info.plist entries to the application after you built it.

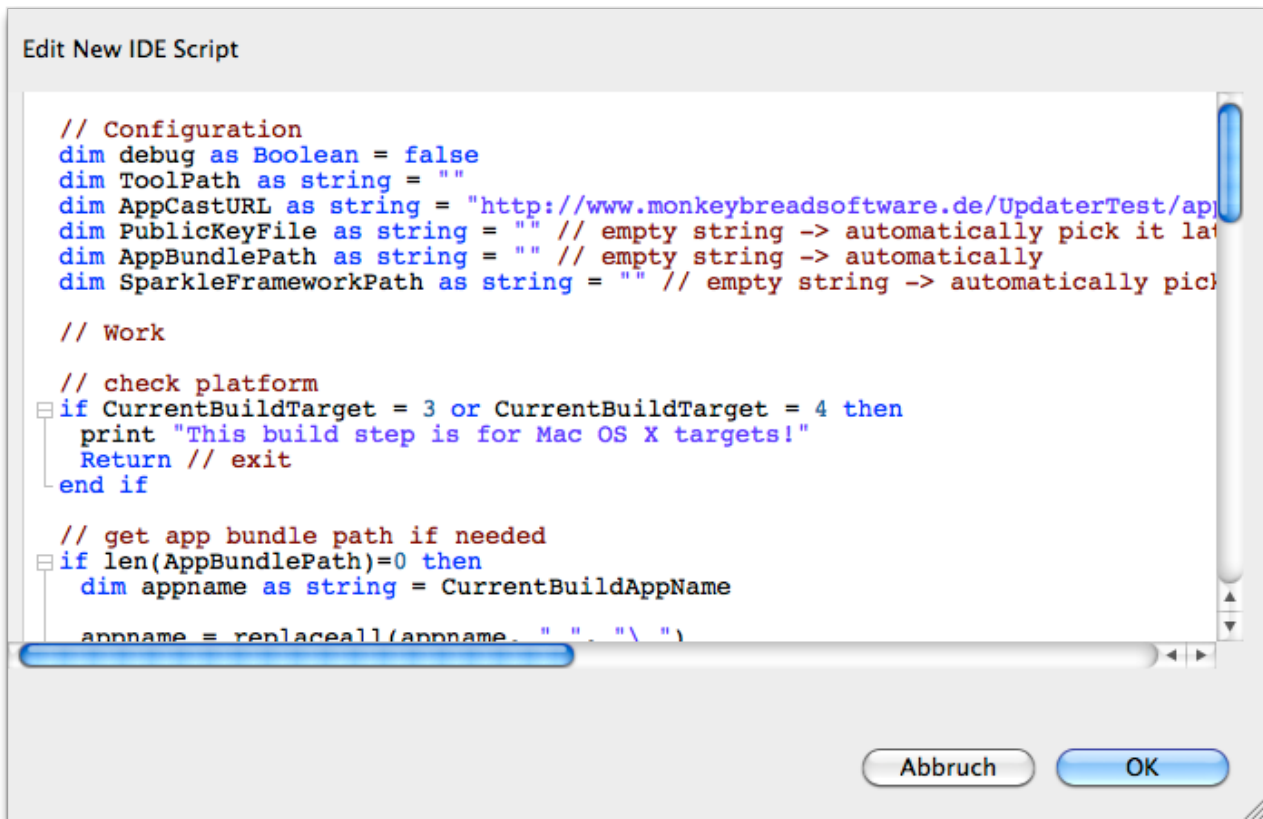
In order to test this script, please follow this steps:

1. Copy the Sparkle.framework and the dsa_pub.pem file to the Build Step folder.
2. Build the "Xojo Build step Desktop.xojo_binary_project" project.
3. Open the Test Project. Inside you see the RunIDEScriptStep build step for Mac OS X
4. You can build it and the final application has the Sparkle files added.

You'll see that Xojo will run our build step tool and add the public key file as well as the plist entry. If no error occurred, you will see no message. We only display error message in case of an error.



The Build Script has a configuration part on the top where you can enter the information you need:



```

// Configuration
dim debug as Boolean = false
dim ToolPath as string = ""
dim AppCastURL as string = "http://www.monkeybreadsoftware.de/UpdaterTest/appcast.xml"
dim PublicKeyFile as string = "" // empty string -> automatically pick it later
dim AppBundlePath as string = "" // empty string -> automatically pick it later
dim SparkleFrameworkPath as string = "" // empty string -> automatically pick it later

// Work

// check platform
if CurrentBuildTarget = 3 or CurrentBuildTarget = 4 then
    print "This build step is for Mac OS X targets!"
    Return // exit
end if

// get app bundle path if needed
if len(AppBundlePath)=0 then
    dim appname as string = CurrentBuildAppName
    appname = replaceall(appname, " ", "\ ")

```

`dim debug as Boolean = false`
this switch enables a few message dialogs to show variables.

`dim ToolPath as string = ""`
The path to the Build Step command line tool. If you leave this value blank, the script uses the default location which is „Build\ Step\Builds\ -\ Xoyo\ Build\ step\ Console.xoyo_binary_project\Mac\ OS\ X\ \Intel\Xoyo\ Build\ step\Xoyo\ Build\ step“ relative to the project folder.

`dim AppCastURL as string = "http://www.domain.test/yourapp/appcast.xml"`
This URL defines the location of your appcast.xml file. Make sure each project has it's own URL. You can use any file name and any file extension as you like.

`dim PublicKeyFile as string = ""`
This option defines the location of the public key file. If this string is empty, the file should be named `dsa_pub.pem` and be located in the project folder.

`dim AppBundlePath as string = ""`
This option defines the location of the build application. If this string is empty, the script asks Xojo for it's location. We recommend that you leave this option empty to take the automatic way.

Be aware that we have a `replaceall` in the script to replace spaces in the application names. Other special characters may need also a replacement.

`dim SparkleFrameworkPath as string = ""`
This option defines the location of the Sparkle.Framework file. If this string is empty, the file should be named `Sparkle.Framework` and be located in the project folder.

If everything works and Xojo perform the script without error, you should only notice it by seeing the files in place and the keys in the info.plist file.

Our Build Script

```
// Configuration
dim debug as Boolean = false
dim ToolPath as string = ""
dim AppCastURL as string = "http://www.monkeybreadsoftware.de/UpdaterTest/
appcast.xml"
dim PublicKeyFile as string = "" // empty string -> automatically pick it
later from the project folder
dim AppBundlePath as string = "" // empty string -> automatically
dim SparkleFrameworkPath as string = "" // empty string -> automatically pick
it later from the project folder

// Work

// check platform
if CurrentBuildTarget = 3 or CurrentBuildTarget = 4 then
    print "This build step is for Mac OS X targets!"
    Return // exit
end if

// get app bundle path if needed
if len(AppBundlePath)=0 then
    dim appname as string = CurrentBuildAppName
    appname = replaceall(appname, " ", "\ ")
    AppBundlePath = CurrentBuildLocation + "/" + appname + ".app"
end if

// find project folder in the ProjectShellPath
dim p as integer = instr(ProjectShellPath, "/")
dim q as integer = 1

while p<>0
    q = p

    p = instr(p+1, ProjectShellPath, "/")
wend

dim ProjectFolderShellPath as string = left(ProjectShellPath, q)

// find public key file if needed
if len(PublicKeyFile)=0 then
    PublicKeyFile = ProjectFolderShellPath + "dsa_pub.pem"
end if

// find Sparkle framework if needed
if len(SparkleFrameworkPath)=0 then
    SparkleFrameworkPath = ProjectFolderShellPath + "Sparkle.framework"
end if

// find tool path if needed
if len(ToolPath)=0 then
    ToolPath = ProjectFolderShellPath + "Builds\ -\ REAL\ Studio\ Build\
step.rbp\Mac\ OS\ X\ \ (Universal\)/REAL\ Studio\ Build\ step\REAL\ Studio\
Build\ step"
end if

// debug
if debug then
    print "ProjectShellPath: "+ProjectShellPath+endofline+_
```

```

    "CurrentBuildAppName: "+CurrentBuildAppName+endofline+_
    "CurrentBuildLocation: "+CurrentBuildLocation+endofline+_
    "AppCastURL: "+AppCastURL+endofline+_
    "PublicKeyFile: "+PublicKeyFile+endofline+_
    "AppBundlePath: "+AppBundlePath+endofline+_
    "SparkleFrameworkPath: "+SparkleFrameworkPath
end if

// this will do the work
dim command as string = ToolPath+" "+AppCastURL+" "+PublicKeyFile+"
"+AppBundlePath+" "+SparkleFrameworkPath

// run shell command
if debug then
    print command
end if

dim result as string = DoShellCommand(command)
if instr(result, "OK") >= len(result)-5 and not debug then
    // ok
else
    print result
end if

```


32-bit/64-bit transition

Before releasing anything with 64-bit, please consider your update strategy.

Mac OS X

On Mac OS X all Xojo 2016 apps require OS X 10.7 and all Macs running that version can run 64-bit applications. You could simply have installer deliver a 64-bit version when you switch from one version to the next one.

But your app may rely on plugins or extensions which may not be available for 64-bit. Or user uses an older version of the app on a 32-bit version of Mac OS X.

The easier way is to have a different appcast URL for different branches of updates.

You can also use `minimumSystemVersion` key in the appcast to have two items there and one marked as `minimumSystemVersion` of 10.7 to make sure the 64-bit version is only delivered to 10.7 and newer.

Windows

For Windows, some users may have a 64-bit version and some a 32-bit version. You can use `SystemInformationMBS.Is64bitWindows` function to know if you have a 32-bit Windows app running on a 64-bit Windows version.

In general we recommend to have two appcast files, one for 32-bit and one for 64-bit. This way people can continue with their bit number until they reinstall.

Because if users have a 32-bit version, they may want to stay in 32-bit app as they may depend on drivers for 32-bit for video devices, scanners or printers.

If you have one installer for both 32/64-bit, you can use one appcast and have everyone load the same updater.

Changes

Tip: If you want to update your existing code with new release, you'd best compare projects with Arbed (<http://www.tempel.org/Arbed>) and copy modifications to new project.

Version 1.0

- First public release.

Changes for Version 1.1

- The `kYouhavethecurrentversion` text was not showed when the current version is the version on the server.
- Added `Properties FoundCurrentVersion` and `FoundUpdate to UpdateEngine`.
- `UpdateEngine.WalkItem` changed like this to detect if the current version is the version on the server.
- In `AppCast Project` in the `Window1.LoadPref` method we added a check to avoid a `NilObjectException`.
- The `VersionIsNewer` method was changed in big parts as it failed with a few version numbers.
- And we got a few more Testcases to `VersionIsNewer`.
- The `AppCast` creator now runs even if not preferences file is found.

Changes for Version 1.2

- Changed the language handling for localization. You need to change your appcast as we now have `sparkle:releaseNotesLink` tags with language attributes to handle the localization of the descriptions.
- Added italian translation.
- Added `#pragma BreakOnExceptions off` to `GotXML` method.
- Added `PatchMacApp` in a console variation.
- The registry key we use to store the preferences is no longer based on the domain name, but on the whole URL of the appcast. So you can use the `Updater Kit` in more than one app.

Changes for Version 1.3

- Added `UpdaterEngine.GetUpdater` to access the updater directly.
- Added `UpdaterEngine.automaticallyCheckForUpdates` property

Changes for Version 1.4

- Updated Windows dialogs to look more like Sparkle.
- Added more localization.
- Added `TestWindowsOnMac` property so you can test Windows Update on Mac.
- Added `ApplicationName` parameter for the `Init` method so you can pass a localized application name. If you pass no name, we take it from the application file name.
- Added support for redirects on the `appcast.xml` and the installer download. Up to 5 redirects are followed before an error is displayed.
- Added new `Build Step` tool and example.

- Changed placement of updater window to be centered on main screen.
- Improved application size by using more conditional compiling.

Changes for Version 1.5

- We now detect older version and tell user that he has the current version. This avoids showing an error for users who have a newer version (e.g. a beta version).
- Updated to Real Studio 2012
- Fixed kYouhavethecurrentversion to show name of app.

Changes for Version 1.6

- Updated for Xojo.
- Fixed Patch App to use Intel instead of Universal for the build folder name.
- Fixed UpdaterEngine to better detect if version is newer or older.
- Updated Sparkle to version 1.6.1

Changes for Version 2.0

- Sparkle now moved to Frameworks folder for better code signing compatibility.
- Added CURL support for downloading on Windows. This allows client SSL certificates, proxies and HTTP/1.1.
- Fixed minor bugs with load/save preferences in case you run on Linux and we want to avoid RegistryAccessErrorException.
- Updated to Sparkle 1.13.1
- Can work for 64-bit targets, too.
- Can download Linux installer, too.

Changes for Version 2.1

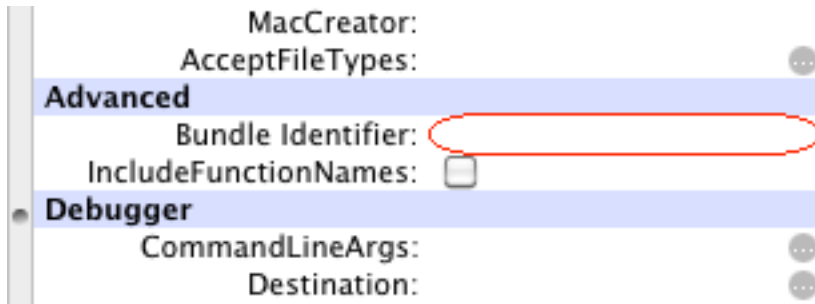
- Fixed timeout property for CURL.
- Updated for Xojo 2017.

Troubleshooting

No update is found

- Is the AppCast URL correct in the info.pAutoSendBugreportAfterDelaylist (Mac) or the call to the init function (Windows)?
- Does your application have a valid bundle identifier?

On Mac no updates are found if the bundle identifier is missing. You can set it in the app class:



- Is the version number in app.shortversion really newer than the one in the AppCast.xml file?
- Is the date correct in the AppCast.xml file?

Update verification fails

- Do you have the public key in your application bundle? (Mac)
- Does the length in the AppCast.xml file match the length of the zip file on Mac and the exe file on Windows?
- Does the checksum match for Windows?
- Are you sure you use your private key for signing and not the sample key?

Requirements

You need the MBS Xojo Plugins. This kit is only available to customers which have a current MBS Xojo Complete license so they can use it.

The minimum plugin set needed to run our updater example project is currently:

- Cocoa
- CocoaBase
- CocoaExtras
- MacOSX
- Main
- Network
- Util
- Win

Also we recommend to use Xojo version 2013 or newer.

License

Summary:

- You need one license of the Updater Kit for each Xojo developer.
- You agree not to share the Updater Kit or use someone else's Updater Kit copy.

Christian Schmitz Software GmbH, of Nickenich Germany is the owner, developer and sole copyright holder of this product, which is licensed -not sold- to you on a non-exclusive basis.

You agree not to share your MBS Xojo Updater Kit with anyone.

You may transfer your license to another person only after receiving written authorization from Christian Schmitz Software GmbH and only if the recipient agrees to be bound by the terms of this agreement.

Christian Schmitz Software GmbH reserves the right to cancel the license key(s) of any user who Christian Schmitz Software GmbH determines is in violation of this agreement.

THE WARRANTIES IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE IS PROVIDED "AS IS" AND Christian Schmitz Software GmbH DISCLAIMS AND EXCLUDES ALL OTHER WARRANTIES. IN NO EVENT WILL Christian Schmitz Software GmbH BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, EVEN IF WE HAVE KNOWLEDGE OF THE POTENTIAL LOSS OR DAMAGE.

If you are located in Germany this agreement is subject to the laws of Germany. If you are located outside Germany local law may apply. Some states do not allow the exclusion of warranties, so the above exclusion may not apply to you.

Christian Schmitz Software GmbH does not charge royalties or deployment fees for Xojo applications.

Access to updates is included for one year. After that time you can order an update or keep using the old version you have.