



# SPARKLE

*Free updates for one and all*

Version 1.1

<http://www.andymatuschak.org/pages/sparkle>  
[andy@andymatuschak.org](mailto:andy@andymatuschak.org)

# TABLE OF CONTENTS

## *I. Introduction*

## *II. Using Sparkle in a Project—the Basics*

## *III. Next Steps*

*A. Adding a Check on Startup Preference*

*B. Scheduling a Periodic Check*

*C. Improving Release Notes*

*D. Checking for Updates Programmatically*

*E. Offering Multiple Appcasts*

*F. Offering Localized Appcasts*

## *IV. Advanced Functionality*

*A. Providing MD5 Sums*

*B. Providing DSA Signatures*

*C. Fully Automatic Updates*

*D. Advanced Version Strings*

*E. Dealing with Registration Dialogs and Other Startup Windows*

*F. Checking to See if Updates Are Available Programmatically*

*G. Performing Actions Before Restart*

*H. Dynamic Release Notes*

*I. Statistics with SparklePlus*

## *V. Conclusion*

## *VI. Appcast Sample*

## *VII. Thanks*

## I. Introduction

Sparkle is a simple module that developers can drop into their apps to get free software update capabilities. It's very easy to set up (especially if you already have an appcast), and it provides many features other solutions don't, like self-updating, release notes viewing, archive support (zip, dmg, tar, tgz, tbz), automatic updates, appcast support, and a nice status interface.

If you'd like to see what Sparkle can do for you, just check out the Demo App.

## II. Using Sparkle in a Project—the Basics

Follow this guide, and you'll have Sparkle up and running in your application in no time at all! Just a few quick steps:

1. **Link the framework:** Drag Sparkle.framework into the Linked Frameworks folder in the Files list of your Xcode project, making sure to check the “Copy items into destination group's folder” box in the resulting sheet.
2. **Deploy the framework:**
  - a. Create a new Copy Files build phase (Project > New Build Phase) and select Frameworks for the destination.
  - b. Switch to the Project page (⌘-0), find your target in the files list, and reveal its contents by clicking the disclosure triangle next to it.
  - c. Drag Sparkle.framework from the Linked Frameworks folder to the new Copy Files phase.
3. **Instantiate the updater:**
  - a. Open MainMenu.nib in Interface Builder.
  - b. Click the Classes tab.
  - c. Go to Classes > Read Files..., navigate to your project directory, then import Sparkle.framework/Headers/SUUpdater.h. SUUpdater should now appear in the class list.
  - d. Select SUUpdater and go to Classes > Instantiate SUUpdater.
  - e. Create a “Check for updates...” menu item in the application menu and connect it to checkForUpdates: in SUUpdater.

#### 4. **Make an appcast:**

- a. Sparkle uses *appcasts* for update information. If you already have one, you can skip this section.
- b. Appcasts are simple: they are RSS feeds with enclosures around software. For Sparkle, these enclosures contain updates to your application. You can make RSS feeds by hand or with the help of another application (Feeder is a nice one).
- c. Whenever you release a new version of your application, compress the .app into an archive (zip, tar, tbz, tgz, or dmg), name it with the format: *YourApp\_VersionNumber.zip*. For example: Pixen\_2.4.zip. Then add a new item to your appcast enclosing that archive.
- d. The description property (or body) should contain release notes for the update (use HTML, but escape < and > with &lt; and &gt; or CDATA tags) or the URL of a web page with the release notes to display. If you don't want to display release notes, set **SUShowReleaseNotes** to <false/> (or No in Property List Editor) in Info.plist.
- e. Sparkle adds some new attributes and properties to the RSS standard (more on those later). To use these features, you'll need to add a URI to your <rss> tag like this:

```
<rss version="2.0" xmlns:sparkle="http://www.andymatuschak.org/xml-namespaces/sparkle">
```

- f. See the *Appcast Sample* section later in this document for a sample feed you can use.

#### 5. **Enter your appcast's URL:** open your target's Info.plist and add an *SUFeedURL* key set to the URL of your appcast:

```
<key>SUFeedURL</key>  
<string>http://www.yourdomain.com/app/example.xml</string>
```

And that's it! You're done! Build and run, and Sparkle should be up and running. You might want to create a test update in your appcast to see how everything works.

## III. Next Steps

Okay, so you've got Sparkle installed—that's great—but it can do a lot more than it's doing right now. This section will walk you through some of the more basic features that are easy to implement in your application.

## A. ADDING A CHECK ON STARTUP PREFERENCE

It's not really fair to ask the user for a preference once and then to never let him change it. You should add a check box for checking on startup to your preferences panel. It couldn't be easier: just add a check box and bind its value to the **SUCheckAtStartup** user defaults key.

## B. SCHEDULING A PERIODIC CHECK

If your program is likely to be running all the time, just checking on startup wouldn't be that useful. Fortunately, you can schedule a periodic check. To do this, add a key for **SUScheduledCheckInterval** to Info.plist, specifying the interval in seconds. For instance, one a day would be  $60 * 60 * 24 = 86400$ .

You might make a menu in your preferences to let the user specify this interval; just update **SUScheduledCheckInterval** in the user defaults with the user's choice—a setting there will override one in Info.plist. You'll need to let Sparkle know when the interval changes, though: just call *scheduleCheckWithInterval:* on your SUUpdater (it'll cancel the old one).

If you have a check interval specified, the **SUCheckAtStartup** key is overridden. Instead, Sparkle will only check on startup if it's been longer than your interval since the last check. If the check interval key is in Info.plist, Sparkle will not ask the user if he'd like to check for updates on startup the first time the application is launched.

This value is also used for the Remind Me Later interval, though if one isn't specified, Sparkle will remind the user half an hour after clicking the button.

## C. IMPROVING RELEASE NOTES

The release notes viewer is actually a full-fledged WebView, so you can use advanced CSS and so on in your release notes to make them extra pretty.

I've included some templates in the Extras that can help you make some really snazzy release notes. They're based on Apple's help books.

If you'd like to use HTML directly in your notes, it can be a hassle to properly escape the HTML to conform with the RSS spec. To avoid this, you can just use CDATA tags:

```
<description><![CDATA[
    <ul>
        <li>Some <b>task!</b></li>
    </ul>
]]></description>
```

However, to avoid the tremendous overhead of including the release notes directly in your appcast (and thus having to transmit them every time a user's app checks for updates), you can instead just refer to a web page on your server. Just set the description of an item to contain that URL (and only that URL).

If you'd like to keep your feeds human-readable in RSS readers, you probably wouldn't want to have your description be a URL. You can override the description with a **sparkle:releaseNotesLink** key in an appcast item like this:

```
<sparkle:releaseNotesLink>http://mydomain.org/myapp_3.1.html</sparkle:releaseNotesLink>
```

#### D. CHECKING FOR UPDATES PROGRAMMATICALLY

If you'd like to add a button to check for updates to some part of your interface, all you need to do is call *checkForUpdates*: on your SUUpdater in the action method (or hook it up directly).

If the check is automated or programmatic, use *checkForUpdatesInBackground*; it won't report errors like "can't connect to server" or "no update available."

#### E. OFFERING MULTIPLE APPCASTS

If you run a beta program, you'd probably like to offer a choice between beta and stable appcasts. Sparkle can make that easy. Just make a menu for this and whenever its value is changed, update the **SUFeedURL** key in the user defaults—whatever's set there will override the value in your default.

#### F. OFFERING LOCALIZED APPCASTS

If you've published an app with extremely wide distribution and localizations in many languages, you might like to localize your appcasts. You can do this by setting the **SUFeedURL** key to the URL of a localized feed in the InfoPlist.strings file in your locale directories. For example, you could set this key to `http://mydomain.org/myapp_fr.xml` in `French.lproj/InfoPlist.strings`. This value will override the one in `Info.plist`.

## IV. Advanced Functionality

There's more, but you might not care. Just read the section headings and see if you're interested in any of these.

## A. PROVIDING MD5 SUMS

If you're looking for security, skip this: info on DSA signing is below. This is more for error-checking: making sure the file's not corrupt before installing it. You can provide an MD5 sum for Sparkle to check against using the **sparkle:md5Sum** attribute on `<enclosure>`. For example:

```
<enclosure sparkle:md5Sum="1fdc47479c8559a00bc58ebce4d84944" url="http://
you.org/yourapp_2.0.zip" length="12345" type="application/octet-stream"/>
```

This value was generated with the console *md5* tool. Just pass it your filename, and it'll spit out a sum.

## B. PROVIDING DSA SIGNATURES

So you're paranoid, but rightfully so: security's a big issue these days. To be *absolutely sure* the archive is what it's supposed to be, Sparkle supports DSA signatures. More specifically, base64-encoded DSS<sub>I</sub> signatures of a SHA<sub>I</sub> hash.

The following is a walkthrough on how to generate keys and sign a file. If you know what you're doing, skip this.

DSA works with two keys: a private one and a public one. You distribute the public one in your app (Info.plist) and sign things with the private one. Your app can verify signatures with just the public signature but *can't create new signatures*. And so long as you use enough bits, you're pretty much guaranteed secure.

To make keys, type these commands into your terminal:

```
openssl dsaparam 2048 < /dev/urandom > dsaparam.pem
openssl gendsa dsaparam.pem -out dsa_priv.pem
openssl dsa -in dsa_priv.pem -pubout -out dsa_pub.pem
```

Now you've got two key files: `dsa_priv.pem` (private) and `dsa_pub.pem` (public). You can delete `dsaparam.pem`. Keep `dsa_priv.pem` secret and back it up—make sure you keep it safe, because if you lose it, you'll never be able to update your current userbase.

Whenever you want to sign a file, you'll use this lengthy command. It first takes an SHA<sub>I</sub> hash of your file, then encrypts it with your private key, then encodes it to base64:

```
openssl dgst -sha1 -binary < FILENAME_HERE | openssl dgst -dss1 -sign
dsa_priv.pem | openssl enc -base64
```

**</dsa tutorial>**

You'll need to set the **SUPublicDSAKey** key in Info.plist to the contents of your **public** key (dsa\_pub.pem). You'll need to use a text editor for this, as Property List Editor doesn't like multi-line strings.

Once you're ready to use DSA in your app, you need to enable it by setting the **SUExpectsDSASignature** key to <true/> (or Yes in PLE). *Once this is on, DSA signatures will be required on all appcast items.* This is to prevent in-between attacks that just remove the signatures from the feed.

Then, for a release, set the signature with the **sparkle:dsaSignature** attribute on <enclosure>:

```
<enclosure sparkle:dsaSignature="MC0CFBfeCa1JyW30nbkBwainOzrN6EQuAh="
url="http://you.org/yourapp_2.0.zip" length="12345" type="application/octet-
stream"/>
```

Be careful with licenses for distribution in certain countries, though: export of strong encryption technologies may be restricted.

## C. FULLY AUTOMATIC UPDATES

Sparkle now offers a new feature: fully automatic updates. When this feature is on, the update panel displays a new check box: "Automatically download and install updates in the future." It does what it says it does. When a new update has been installed, it informs the user and offers to relaunch the app, but he can dismiss the dialog and relaunch at his leisure.

This feature is available (for security reasons) unless the DSA signing is enabled with the **SUExpectsDSASignature** key; for more information on that, see IV.B.

You can explicitly disable this feature by setting the **SUAllowsAutomaticUpdates** key to <false/> (or No in PLE) in Info.plist. But please don't release an app with this on until you've tested thoroughly and are satisfied it works.

## D. ADVANCED VERSION STRINGS

I'll be the first to admit that the naming convention of *YourApp\_VerNo.zip* is lame and brittle. So if you'd like to use fancy version numbers (potentially with underscores) or even if you'd just like to make your system a little less brittle, know that you can name your files whatever you want and then specify the version with the **sparkle:version** attribute on <enclosure> like this:

```
<enclosure sparkle:version="3.0_rc2" url="http://you.org/something.zip"
length="12345" type="application/octet-stream"/>
```



Furthermore, if you use an internal build number for your app with `CFBundleVersion` but set a display string for the version with `CFBundleShortVersionString`, Sparkle also supports this behavior. To specify a short version string for an update (which you should do to be consistent if you use this key), you can use the **sparkle:shortVersionString** attribute on *<enclosure>* just as above.

## E. DEALING WITH REGISTRATION DIALOGS AND OTHER STARTUP WINDOWS

If your application displays other introductory or “welcome”-ish windows when you start it up for the first time, it would be jarring and would create a poor first impression to have Sparkle’s check-on-startup dialog jump up, too.

In this case, you can take over the duty of presenting this option to the user by setting the **SUCheckAtStartup** key to `<false/>` (or No in PLE) in Info.plist. Then add a checkbox to the last pane of your registration wizard or whatever that’s bound to this key in the user defaults.

## F. CHECKING TO SEE IF UPDATES ARE AVAILABLE PROGRAMMATICALLY

If you need to programmatically check to see if an update is available (but without actually offering to perform the update), Sparkle provides a special class for just that purpose.

**SUStatusChecker** is an `SUUpdater` subclass with a nice factory method: +

*statusCheckerForDelegate*: Make sure to retain it, since it’ll have a retain count of 0.

Just call that with your relevant controller object as delegate, then call *checkForUpdatesInBackground*. Once `SUStatusChecker`’s gotten answer from the server, it will call this method on your delegate:

```
- (void)statusChecker:(SUStatusChecker *)statusChecker foundVersion:(NSString *)versionString isNewVersion:(BOOL)isNewVersion
```

`versionString` will be nil and `isNewVersion` will be NO if version checking fails.

Here’s some sample code:

```
- (void)startCheck
{
    [[[SUStatusChecker statusCheckerForDelegate:self] retain]
    checkForUpdatesInBackground];
}

- (void)statusChecker:(SUStatusChecker *)statusChecker foundVersion:(NSString
```

```

*)versionString isNewVersion:(BOOL)isNewVersion
{
    if (!versionString)
    {
        NSLog(@"Couldn't get new version information!");
        return;
    }
    NSLog(@"Newest version: %@", versionString);
    NSLog(@"Is newest? %@", isNewVersion ? @"Yes" : @"No");
}

```

## G. PERFORMING ACTIONS BEFORE RESTART

If you normally do some big, time-consuming action before quitting or need to clean something up before the app shuts down, Sparkle can help. It sends out a **SUUpdaterWillRestartNotification** before restarting; just observe this and perform the necessary actions. You'll need to `#import <Sparkle/Sparkle.h>` to get access to this constant.

## H. DYNAMIC RELEASE NOTES

Jacob Godwin-Jones wrote an excellent article on dynamically generating release notes for your Sparkle users on your webserver. With this solution, users upgrading from 2.0 to 2.2 see changes in both updates. To learn how to implement it, read Jacob's article at <http://www.likethought.com/blog/2006/05/29/dynamic-sparkle-release-notes/>.

## I. STATISTICS WITH SPARKLEPLUS

Would you like to know more about your users? What software and hardware configurations they run? So would Tom Harrington! He wrote a patch to Sparkle that reports (at the user's option) quite a lot of system information. I didn't want it in the core distribution—since this is such a widely-used framework, there are ethical implications—so Tom forked the patch into SparklePlus, which you can grab at <http://ironcoder.org/blog/2006/06/14/sparkle-plus>.

## V. Conclusion

I hope that Sparkle will save you—and your users—some time and headaches. If it works well for you, please let a friend know about it! Let's put an end to manual updates together.

This software is under an MIT License (see *License.txt*), so you can basically do whatever you want with it. A note of thanks in an about box would be nice, though. Also, if you make any useful changes or additions, I'd love to get a copy of the patch. Finally, if you release an app

that uses Sparkle, please let me know so I can add you to a list on Sparkle's page! Also enclosed is a little Sparkle button you can use to spread the word.

## VI. Appcast Sample

```
<?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>Some App Changelog</title>
    <link>http://mysite.org/appcast.xml</link>
    <description>Most recent changes with links to updates.</description>
    <language>en</language>
    <item>
      <title>Version 2.0 (2 bugs fixed; 3 new features)</title>
      <description>http://mysite.org/noteson2.0.html</description>
      <pubDate>Wed, 09 Jan 2006 19:20:11 +0000</pubDate>
      <enclosure url="http://mysite.org/files/myapp_2.0.zip" length="1600000"
type="application/octet-stream"/>
    </item>
  </channel>
</rss>
```

## VII. Thanks

Many thanks to Allan Odgaard, who provided code samples that helped me with tar extraction, DSA validation, application installation and authentication.

Thanks to Jeff Marlow for the lovely application icon.

Thanks to codepoet for a much-improved zip extraction implementation.

Thanks to Daniel Wilson for UI criticism and advice.

Thanks to David Kocher and the Cyberduck localizers for providing localizations.

Thanks to David Young for his help with WebKit-based release notes.

Thanks to Evan Schoenberg for his SUStatusChecker patch and some minor bugfixes; also for hooking me up with some additional localizers.

Thanks to M. Uli Kusterer, whose UKNiftyFeatures module inspired this project.

Thanks to Joe Osborn for his refactoring help.

Thanks to Brent Simmons, whose RSS reader class was modified and used for appcast support. His class (not Sparkle itself) is licensed as follows:

Copyright (c) 2002, Brent Simmons  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of ranchero.com or Brent Simmons nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This project uses software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org>). This toolkit is licensed (at length) as follows:

```
/* =====  
* Copyright (c) 1998-2005 The OpenSSL Project. All rights reserved.  
*  
* Redistribution and use in source and binary forms, with or without  
* modification, are permitted provided that the following conditions  
* are met:  
*  
* 1. Redistributions of source code must retain the above copyright  
* notice, this list of conditions and the following disclaimer.  
*  
* 2. Redistributions in binary form must reproduce the above copyright  
* notice, this list of conditions and the following disclaimer in  
* the documentation and/or other materials provided with the  
* distribution.  
*  
* 3. All advertising materials mentioning features or use of this  
* software must display the following acknowledgment:  
* "This product includes software developed by the OpenSSL Project  
* for use in the OpenSSL Toolkit. (http://www.openssl.org/)"  
*  
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to  
* endorse or promote products derived from this software without  
* prior written permission. For written permission, please contact  
* openssl-core@openssl.org.  
*  
* 5. Products derived from this software may not be called "OpenSSL"  
* nor may "OpenSSL" appear in their names without prior written  
* permission of the OpenSSL Project.  
*  
* 6. Redistributions of any form whatsoever must retain the following
```

\* acknowledgment:  
 \* "This product includes software developed by the OpenSSL Project  
 \* for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"  
 \*  
 \* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY  
 \* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
 \* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR  
 \* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR  
 \* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  
 \* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT  
 \* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;  
 \* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)  
 \* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,  
 \* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)  
 \* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED  
 \* OF THE POSSIBILITY OF SUCH DAMAGE.  
 \* =====  
 \*  
 \* This product includes cryptographic software written by Eric Young  
 \* (eay@cryptsoft.com). This product includes software written by Tim  
 \* Hudson (tjh@cryptsoft.com).  
 \*  
 \*/

#### Original SSLeay License

/\* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)  
 \* All rights reserved.  
 \*  
 \* This package is an SSL implementation written  
 \* by Eric Young (eay@cryptsoft.com).  
 \* The implementation was written so as to conform with Netscapes SSL.  
 \*  
 \* This library is free for commercial and non-commercial use as long as  
 \* the following conditions are aheared to. The following conditions  
 \* apply to all code found in this distribution, be it the RC4, RSA,  
 \* lhash, DES, etc., code; not just the SSL code. The SSL documentation  
 \* included with this distribution is covered by the same copyright terms  
 \* except that the holder is Tim Hudson (tjh@cryptsoft.com).  
 \*  
 \* Copyright remains Eric Young's, and as such any Copyright notices in  
 \* the code are not to be removed.  
 \* If this package is used in a product, Eric Young should be given attribution  
 \* as the author of the parts of the library used.  
 \* This can be in the form of a textual message at program startup or  
 \* in documentation (online or textual) provided with the package.  
 \*  
 \* Redistribution and use in source and binary forms, with or without  
 \* modification, are permitted provided that the following conditions  
 \* are met:  
 \* 1. Redistributions of source code must retain the copyright  
 \* notice, this list of conditions and the following disclaimer.  
 \* 2. Redistributions in binary form must reproduce the above copyright  
 \* notice, this list of conditions and the following disclaimer in the  
 \* documentation and/or other materials provided with the distribution.  
 \* 3. All advertising materials mentioning features or use of this software  
 \* must display the following acknowledgement:  
 \* "This product includes cryptographic software written by  
 \* Eric Young (eay@cryptsoft.com)"

\* The word 'cryptographic' can be left out if the routines from the library  
\* being used are not cryptographic related :-).  
\* 4. If you include any Windows specific code (or a derivative thereof) from  
\* the apps directory (application code) you must include an acknowledgement:  
\* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"  
\*  
\* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND  
\* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
\* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE  
\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL  
\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS  
\* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)  
\* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT  
\* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY  
\* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF  
\* SUCH DAMAGE.  
\*  
\* The licence and distribution terms for any publically available version or  
\* derivative of this code cannot be changed. i.e. this code cannot simply be  
\* copied and put under another distribution licence  
\* [including the GNU Public Licence.]  
\*/