Apple® IIGS

Check File
PIF Update Checker
Manual

Dedicated to the memories of Joe Kohn 1947-2010
and Ryan Suenega 1967-2011

Credits

My thanks go Eric Shepherd, for his original suggestion that I create a FinderExtra using the ideas within Versions, and to Chris Vavruska for his extensive help, beta testing, and many suggestions.

Contents

Preface

Welcome to Check File

Introduction

Check File is a PIF file that independently duplicates the function of the 'Check File' menu item within the **Versions** updates checking application.

Check File works on a single file, or multiple files, but for full support, it relies on the target application having the rUpdateInfo resource present. (See Page 11)

Check File will let you quickly check if a file is up-to-date, and if found, display its associated .inf information file, and also if it is available, download the latest version of the archive from its web host.

Check File supports an IPC call, that allows applications to automatically check for updates to themselves when they start up, and even checks itself at boot.

For a fuller scan of your disks for any applications that may be out-of-date, please use the full **Versions** application.

If your favourite applications do not have the rUpdateInfo resource, then you should ask their author to add the resource to their next update, as with the resource added, **Check File** can then check for updates without the author needing to use the full **Versions** update system.

Get **Versions**, and also check out my other contributions to the IIgs world on my web site:

http://speccie.uk

Check File Installation

Requirements

Remove any existing copies of **CheckFile** from the FinderExtras folder, and drop the new **CheckFile** from the archive either onto the System folder icon, or into the System.Setup folder within the System folder. You will need to reboot for Check File to become active.

Check File requires the Marinetti TCP/IP stack to be active, and the HTML Tool (Tool130) to be installed. The Check File menu item will effectively be inactive if these are not present:

To obtain Check File, Versions, or the HTML Tool, and any of my other software:

http://speccie.uk

To obtain the latest version of the Marinetti TCP/IP stack:

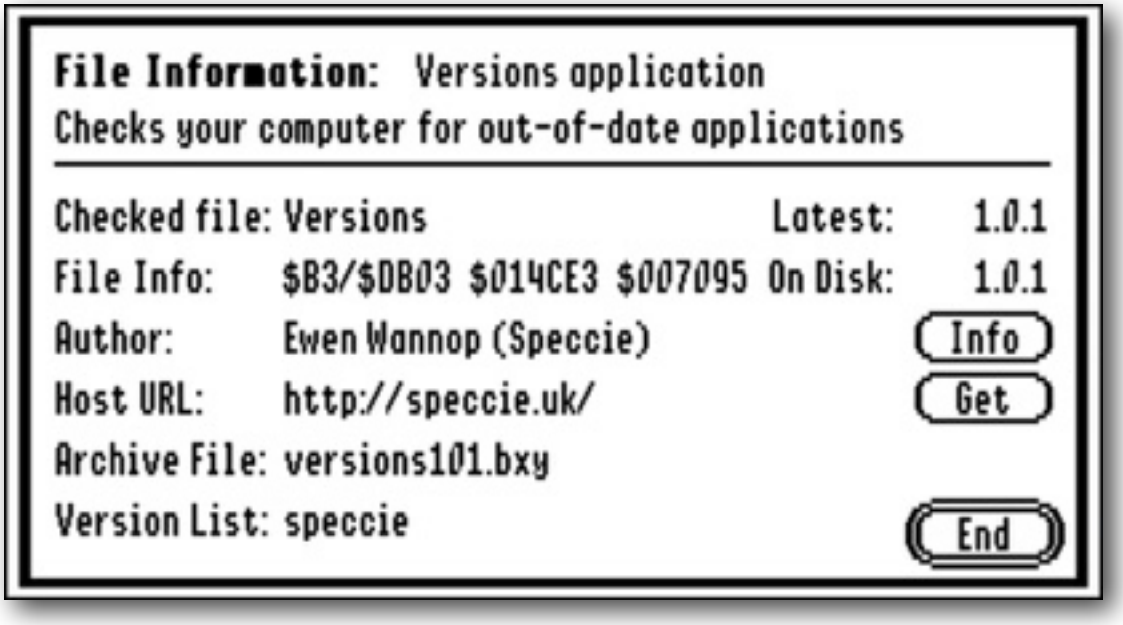
http://www.apple2.org/marinetti/

To find out about, and order an Uthernet card:

http://www.a2retrosystems.com/

Working with Check File

File Selection



How to use Check File

Select one or more applications, or other relevant files from within a Finder window, and choosing '**Check File...**' from the Finder Extras menu. You can select a single file, or multiple files as required. The **Check File** menu item will only be active if a suitable file is selected. Aliases and folders do not count, and will be ignored.

A window opens that has a number of lines of information and buttons. The data you see displayed in the window will depend on whether the file has an rUpdateInfo resource, and whether there are any related .inf or archive files to be found on the related Host server.

Reading from Top to Bottom:

First Line: If an .inf file was found, this shows a description of the selected file extracted from that file.

Next Line: Further description of the selected file extracted from the .inf file. If an rUpdateInfo resource was not present, or there was no .inf file found on the host server, you will see the message (No rUpdateInfo linked Info file found).

Checked File: The filename of the selected file. This will always be displayed.

File Info: The filetype/auxtype, data fork length, and resource fork length, of the selected file. This will always be displayed.

Author: The name of the Author. This will be displayed if an rUpdateInfo resource was found.

Host URL: Path to the downloads folder on the host server. This is only displayed if an rUpdateInfo resource was found.

Archive File: The name of the related archive file. This is only displayed if an rUpdateInfo resource was present. Note that this is the name of the archive file for the current application when it was downloaded. There may be a newer update available. See 'Latest' below.

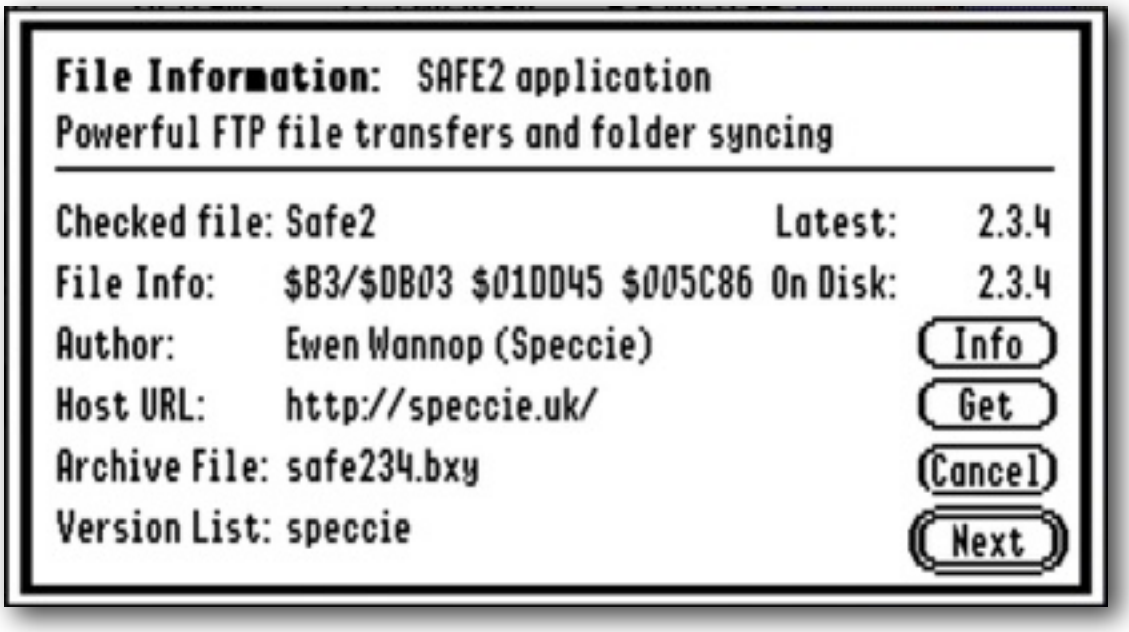
Version List: The name of the Version.List if the author supports the Versions update system. This is only displayed if an rUpdateInfo resource was found.

Latest: The version number of the latest version of the application. This is extracted from a related .inf file, and is only displayed if an rUpdateInfo resource was found.

On Disk: This is the version number of the application you have on disk, and is only displayed if the selected file had a standard rVersion resource.

Note: Many of these fields rely on either an rVersion or rUpdateInfo resource being present. You will always see the file/auxtype, data fork length, and resource fork length even if those resources are not found.

The Buttons:



Info: If a related .inf file was found on the host server, this will display the text from that file in a floating window. This is the same as the Info button in Versions.

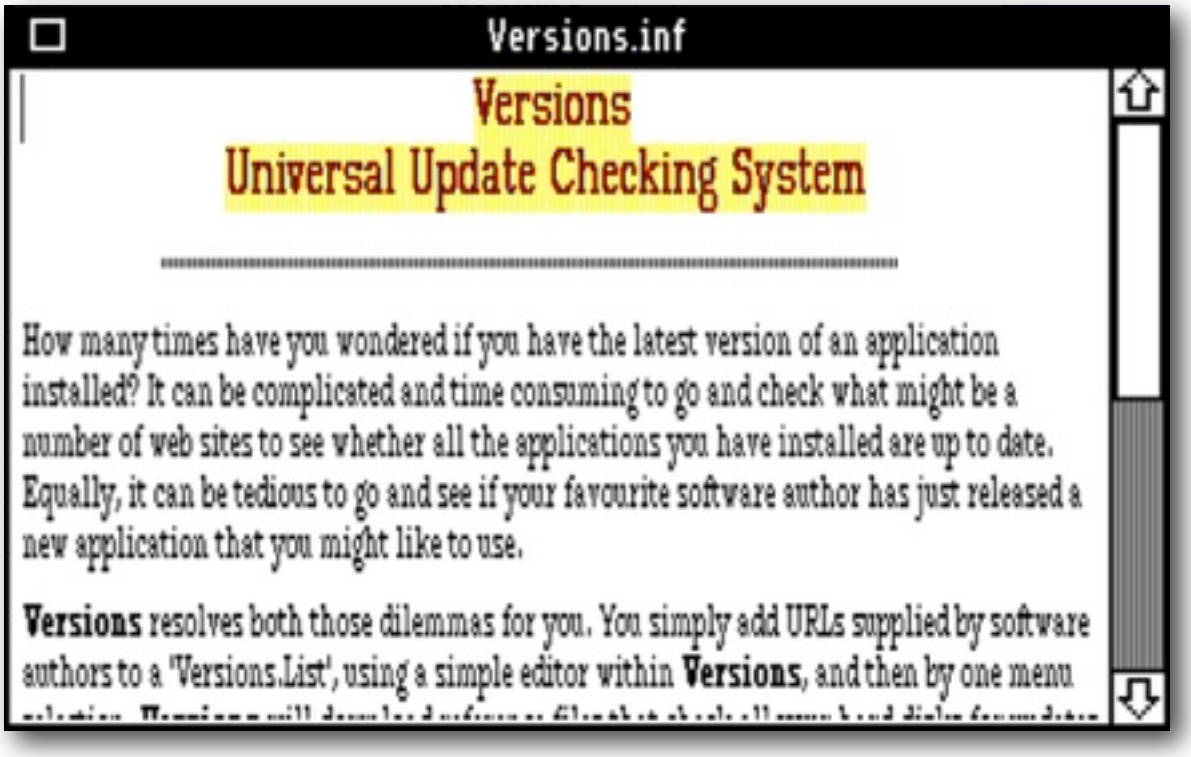
Get: If archive file was found on the server, this will download that file to a folder of your choice. Note that if there was a newer archive displayed in the 'Latest' field, that archive will be downloaded instead of the one indicated in the 'Archive File' field. This means you will always get the newest version of an archive.

Cancel: If you had selected multiple files, this will close the window and let you stop checking files before you have checked them all. Checking for an .inf file can take some time depending on the speed of your network.

Next/End: If only one file has been selected, or you are viewing the last file of a range, this button will say 'End', and will close the window. If you have selected multiple files, this button will say 'Next', and will move you on to the next file you selected.

Note: Check File checks itself for updates at startup. If one is found, you can choose to Download the new version, or to Ignore the self check process.
A marker file CFile.Prefs, will be placed in the System:Preferences folder to show you have clicked Ignore. If you wish self checking of Check File to start again, just delete that file.

The Info Dialog:



Info Files: If a related .inf file was found on the host server, it will display in a floating window. These files are part of the Versions update system, and accompany their downloadable archive file. They may be written as plain text files, or as HTML files.

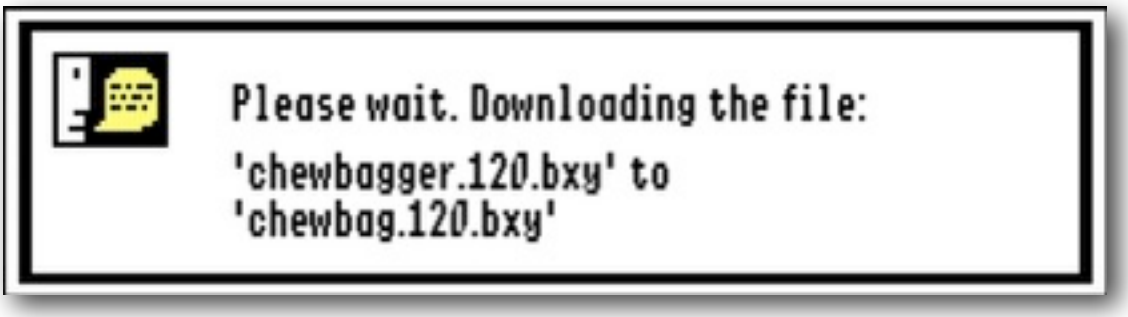
In addition to a description of the file itself, the author may have added other data to the file, such as a Change list, or version releases. The author may also have added a line of text that is used to build the first two lines of the display, display the version number of the latest archive, and show the name of the archive that will be used for downloading. If the .inf file is an HTML display, that line will be added as a tag which will not display on screen, but if the .inf file is plain text, you will see that line displayed.

Note: Developers, please refer to Page 12 for more details on how the tag line is constructed.

Downloading an Archive:

If the 'Get' button is highlighted, and clicked, you will see the standard Put File Dialog, with the name of the archive file that will be download in the 'Safe File as' prompt. You can change this file name if you wish, as some names may be too long for a ProDOS volume.

When you click Save, the file will be downloaded to disk:



If a matching .md5 file is found on the server, that too will be downloaded, replacing the four byte suffix with .md5. However, if you change the name of the file you are saving to disk, and there is no four byte .xxx suffix, the .md5 suffix will be added to the end of the name, truncating the name if necessary to make it fit the volume name system. A ProDOS volume can only handle a 15 byte name, and an HFS volume a 34 byte name.

The rUpdateInfo Resource

From an idea by Eric Shepherd, I have proposed a new Resource format that could be used in future to aid update applications such as Versions.

Edit your 'types.rez' library file to include this resource format:

```
#define rUpdateInfo $DEAD  /* RIP Jerry Garcia */

/*--------------------- rUpdateInfo -------------------------*/
type rUpdateInfo {
    integer = 0;           /* version must be zero */
        longint;           /* Resource ID of string holding app's display name */
        longint;           /* ID of string holding author's name               */
        longint;           /* ID of string URL to the downloads folder         */
        longint;           /* ID of string for downloadable archive name       */
        longint;           /* ID of string for Versions.List data file         */
};
```

Then in your rez file, include these templates and definitions:

```
// --- rPString Templates

// app's display name
resource rPString (PSTR_00002000, $0000) {
       "Versions"
};

// author's name
resource rPString (PSTR_00002001, $0000) {
       "Ewen Wannop (Speccie)"
};

// URL to downloads folder
resource rPString (PSTR_00002002, $0000) {
       "http://speccie.uk/speccie/downloads/"
};

// Name of downloadable archive
resource rPString (PSTR_00002003, $0000) {
       "versions1.0.bxy"
};

// Name of Versions.List file
resource rPString (PSTR_00002004, $0000) {
       "speccie"
};

// --- rUpdateInfo Definitions

resource rUpdateInfo (RUPDATE_00000001, $0000) {
       PSTR_00002000,              // app's display name
       PSTR_00002001,              // author's name
       PSTR_00002002,              // URL to downloads folder
       PSTR_00002003,              // Name of downloadable archive
       PSTR_00002004               // Name of data file for Versions.List
};
```

Using the Resource:

Make sure the resource is defined as $DEAD and has a resource number of $00000001, that all five pStrings are present (their resource numbers are not important, as they will be picked up from the rUpdateInfo resource itself), and if possible have been filled in, or left as null pStrings. It is important that the fields are valid entries, especially 'Name of downloadable archive' and its 'URL to downloads folder', as those will be used for the 'Get' button in Check File to retrieve the archive.

If you have placed a Versions format .inf file onto your server, please make sure it contains either one of these descriptive comment tags. If that tag is not present, Check File cannot fully handle any downloadable archives that are newer that you may have on your server. This is the only way that you can tell the user through Check File that there is an update waiting.

The tags must be created in one of two formats, depending on whether the .inf file is plain text, or formatted as HTML. For instance, for SAFE2, you could use something like this:

```
HTML:   <!--Versions--FTP client--A powerful file transfer FTP
client--2.3.3--safe234.bxy-->

Plain Text:  **--Versions---FTP client--A powerful file transfer FTP
client--2.3.3--safe234.bxy--
```

The <! tag or double asterisk marks the start of the line, and is then followed by '--Versions--', which defines it clearly as a custom tag . There are four segments, each ended by '--' markers. The text of the first segment will show on the first line of the main window (max 34 chars), and describes the application, the second (max 58 chars), shows on the next line, and should describe what the application or file does. The third shows the latest version number of the application in VersionString format, and the fourth is the filename of the downloadable archive on your server.

Note: You can use any punctuation you like within each segment description, except for a double '--', which always marks the end of each segment.

The IPC Call

Check for updates when your application starts:

Many developers may have decided not to use the Versions update checking system, as being perhaps over complicated to set up for their use. Now, with the addition of a single IPC call to Check File at startup, applications can easily check for newer versions of themselves automatically.

To achieve this, the rUpdateInfo resource must be present in the applications resource fork, and a minimally formatted .inf file, along with a downloadable archive, be placed on the web server passed in the rUpdateInfo resource. Marinetti must also be installed and connected to the Internet, though the application itself does not need to start or use TCP/IP itself.

How the IPC call works:

When the IPC call is made, the application passes its Resource ID, and a configurable Input flag to the IPC call. At its simplest, that is all the application needs to do, and Check File will do the rest. The Input flag gives the application choices as to will happen during the call, so the application is always in full control. Please refer to the descriptions below on how that works.

When Check File is called, it retrieves the pStrings from the rUpdateInfo resource of the application, then attempts to retrieve the related .inf file from the target web server named in that resource. If the .inf file is found, it will be parsed to extract the version number of the latest update that is on the server, and also extract the filename of the latest update. It then compares the version number it found against the current version number of the application itself, and if they are different, will show an Alert with Download, Ignore, and OK buttons. By clicking Ignore or OK, or if the .inf or archive files are not found, or the version on the server is the same, Check File will simply return from the call as if nothing had happened. If the Download button is clicked, a Standard File dialog will open to let the user choose where to save the file, and Check File will then download the archive. An associated .md5 file will also be downloaded if one is found.

Typically, this whole checking process takes no more than a second, and if a download is called for, that may only take a few seconds more.

The Input and Output data blocks:

Please refer to the section above for the rUpdateInfo resource to see how to correctly format that resource, and also how to build the required tag for the .inf file that must also be placed on the server along with the downloadable archive.

The Input DataIn block tells Check File what to do when it has been called:

```
Input flag:
0 = Checks for presence of files, does not show an alert
1 = Displays an alert to say an update was found
2 = Displays the Download option if an update and an archive were found
3 = Silently checks for an update, and downloads if one is found
```

The IPC call:

```
         pea   $6200            tellCFToCheckFile
         pea   $8001            stop after one
         PushLong #CFileName
         PushLong #CFDataIn
         PushLong #CFDataOut
         _SendRequest

CFDataIn anop
         dc    i'2'             parameter count = 2 or 3
         dc    i'$0002'         Input flag
         dc    i'0'             this apps ResourceID
         dc    i4'Folder'       path to download folder

CFDataOut anop
         dc    i'0'             recvcount
         dc    i'0'             Foundflag
         dc    i4'0'            Pointer to Version string (pString)
         dc    i4'0'            Pointer to update name (pString)
         dc    i'0'             Returned error of Input flag = 3

CFileName  str   'Speccie~Check.File.PIF~Wannop~'
```

On return from the call, the Foundflag reports what it found:

```
Bits set:
Bit 0 = a matching .inf file was found for the application
Bit 1 = a newer version was found on the server
Bit 2 = the version on the server is the same as the application
Bit 3 = a downloadable update archive was found
Bit 4 = the update archive was not downloaded
Bit 5 = the update archive was downloaded
Bit 6 = the user chose to ignore the update
Bit 7 - file already exists, not downloaded
Bit 14 - GS/OS error during download
Bit 15 = general error, no required files were found
```

A pointer to the version string, and the update file name, that were found in the matching .inf file, are also returned if the .inf file was found.

Strategies

By first setting the Input flag to a suitable value, and copying the apps Resource ID to the data block, the calling application has four options:

Input flag = 0

Check File will not display any Alerts and on return, the application can check the Foundflag to see which files were found, and whether an update is available.

Input flag = 1

If an update is found, Check File will show an Alert, and report that it has found an update. The user will not be able to download an update if one was found found, and the Foundflag will report whether the user clicked the Ignore button. No alert will be shown if there is no new update available.

Input flag = 2

If an update is found, Check File will show an Alert, and report that it has found an update. The user can choose to Download the update, or click Ignore, or OK, and return. The Foundflag will show whether the user clicked the Ignore button. No alert will be shown if there is no new update available.

Input flag = 3

Check File will not display any Alerts, and if an update is found, Check File will silently download the file to the supplied download path. If the file already exists, or the path is invalid, the call will fail, and report any GS/OS error at the end of the DataOut block. It will also set the bits of the Foundflag as appropriate.

In all cases, it is up to the Application to decide what to do after the IPC call. From the Foundflag, it can see if there was an update available, and if so, if the user chose to Ignore it, or just clicked the OK button. If the user chose to Ignore an update, the application may then decide not to make that call again, by setting a flag in its Preferences. This could be whether to let the IPC call be made at Startup, or perhaps make it available as a Menu item choice.

If a matching .inf file was found, even if it was not a newer version, a pString in VersionString format of the archive on the server will be returned, as well as pString holding the name of the update file. If the pStrings are empty, then the Version data and filename were not found in the matching .inf file, or the .inf file itself was not found.

Extras

Problems

Hopefully you will have none, but if you do, and they cannot be answered by reading these notes, please contact me on:

spectrumdaddy@speccie.uk

Other information

Check for the latest version of Marinetti:

http://www.apple2.org/marinetti/

If you do not already know about Spectrum™, please drop by my home pages, and read more. Apart from all the other wonderful things it does, Spectrum™ offers many useful tools for processing files, such as post processing text files that you have received, that may have obstinate formatting.

Spectrum™ is now Freeware, and amongst my other applications, is now available from my web site:

http://speccie.uk

Someone once said to me, 'Spectrum™ does everything!'