# Micalin's Launcher *for 2.0 beta*

## *Users Guide for 2.x*

Document Revision 2.0.0


# Table of Contents

# 1. Introduction, License, Install Notes, and More

## *What does Micalin's Launcher do?*

***Fundamentally this is a collection of folder actions and a management utility for creating and maintaining a Dock folder with aliases to your applications to save you the trouble of doing it yourself.***

More specifically ***Micalin's Launcher*** is an *AppleScript Studio* based application that creates and maintains a folder of aliases to most or all of your installed applications that can be placed in the Dock to function as a "*poor man's*" pop-up launcher, or *cringe,* a Start Menu…*spit*. Folder actions are also installed to automatically update this Dock folder as new applications are added with the ability to direct new applications aliases from any drive folder or folders to any of the Launcher's subfolders via these actions.

## *Why did I make this cheesy thing you ask?*

I originally started ***Micalin's Launcher*** when my girlfriend was transitioning from PC's, and having fits over the lack of a ***Start Menu***. I had been meaning to do something like this since my pre-OS X days when I maintained an Apple Menu folder such aliases, a practice that has continued via a Dock folder in OS X, and this convinced to finally do it. Thinking it would take an hour tops I found out how hard it is to make a simple user aid that doesn't cause more confusion than it prevents. It is not nearly good enough for something to do the adequate thing, the consistent thing, or the logical thing. Instead it must do that thing the user expects it to do in every situation, every time, or it's worthless. As I tried to make the Launcher do what SHE expected my silly project seemed to expand without limit. Since it involved lots of little aspect I used working on it as a way to help me explore AppleScript, and now, AppleScript Studio and it continues to be my learner and experimentation project.

In other words I just made it for fun. I realize other more sophisticated options exist but the point was to see what I could do, and without peaking so much I wouldn't appreciate the answers. The experience has given me a much greater appreciation for what goes into making a piece of software, and for those who write it.

This may be just a big, glorified, kludgy AppleScript but it is *MY* big, glorified, kludgy AppleScript!

***Note that this is a beta document for a beta software release so inaccuracies or omissions are likely. This document was based on the Micalin's Launcher 1.1.2 User's Guide.***

## How does it work?

On "*Install*" ***Micalin's Launcher*** looks through those folders your select to scan on install that can include your entire root, Home, and Classic Applications folders and your root Developer folder and creates aliases to any applications it finds in those folders, classic, carbon, cocoa, dual forked, app packages…you name it. It places these aliases into matched subfolders in a new root folder created in your user Library This folder is the *Applications Launcher* folder, located at *~/Library/Application Support/Applications Launcher Data/Applications Launcher*.

The user can then place this newly root alias folder into the right side of the Dock to act as a pop-up menu . . . we now FORMALLY christen this THE *Launcher (Dock) folder*, just called the *Launcher folder* or *Launcher (Dock) folder* in our Documentation from now on.

A variable number of folder actions are also installed to maintain these alias folder and its subfolders. One, the "Helper" action, watches the Launcher (Dock) folder to relocate applications dropped into it to the applications folder both for user convenience and safety. In addition an Auto-Add action is installed for each Launcher subfolder created and for the main Launcher folder itself that watch selected drive folders for new applications. The Auto-Add actions for a certain Launcher folder can be attached (or linked) any number of regular folders for Auto-Adding of aliases.

Filters are also liberally applied to weed out the various junk applications, such as Electronic Registrations, Setup Wizards, etc, and these filters are automatically appended (if desired) as the user manually weeds out application aliases that they find undesirable. These filters can be expanded or edited individually as well.

## Will this mess up my computer?

I sure hope not. It has never caused problems that I know of and I have taken reasonable precautions to prevent them in the future. In general this application takes a conservative approach to what little it does: 1) It never asks for an *Admin* password and does nothing that would require one, 2) Except for its preference and log files located in the normal places no files or folders outside of the Launcher folder structure that this program creates are ever modified or moved, with the exception of having the Applications folder or Desktop modified when items are moved into it from the Launcher folder.

I like to play with, learn on, and bloat this application because it is relatively harmless and expendable, so hopefully no problems of real significance will ever happen from using it.

However, this the the United States, lawyer capital of the universe, so you might ought to see the standard disclaimer given next…

## *DISCLAIMER*

**In addition to being beta software with the risks tha implies all versions of Micalin's Launcher are released AS IS and have NO WARRANTY expressed, implied, dreamt, or certainly meant (but clearly denied). If if it causes your computer to melt, use poor grammar, lie, cheat, steal, or kick your dog...well...you were warned.**

## *LICENSE*

**Micalin's Launcher is released as "What-hell-ware" (as free software) and can be used and redistributed freely provided this disclaimer and license and, at minimum, the Micalin's Launcher QUICKSTART Guide document accompanies it.**

## *Complaints*

If you like Micalin's Launcher send me an email. If you hate it, send my 6th grade math teacher an email...I have her address here...somewhere...

# Important Install and Use Notes for Version 2.0 beta

This is beta software so bugs and functional gaps remain.

As of build 3364 *(updated to addresses critical Tiger issues)* limited testing in Tiger has been performed and I can confirm the Micalin's Launcher starts, installs, rebuilds, and the folder actions seem to work but significant differences are evident between Panther and Tiger so issues are likely. See ***Tiger Notes*** included as a separate file for more.

Upgrading from pre-2.0 versions is more involved than normal and currently incomplete. The "Upgrader" may confuse one or more of the folders containing "Applications" in the name (although this *may* have been fixed by the Tiger prompted changes). Also the old pre 2.0 icons must be manually updated to the new ones. Instructions are in the QUICKSTART Guide.

The ***Micalin's Launcher*** Help menu item and some Help buttons bring up an internal HTML version of this document in your default web browser.

The terms *Launcher ƒ* and *Launcher (Dock) ƒ* both refer to the folder *Applications Launcher* that you place in your Dock.

If your system is older or a little slow you might get a 'Finder busy' error on install if you are doing something else with the Finder while installing. Just run the program again and choose Rebuild Launcher from the Launcher (Dock) ƒ menu.

By default when you open the *Launcher (Dock) ƒ* by clicking on its Dock icon a folder action immediately closes that folder and launches the **Micalin's Launcher** application. Turn OFF **Simulate Application** to disable this behavior. Turn of the Close Launcher folder option to prevent that folder from automatically closing.

The folder actions are located in your user Library at *~/Library/Scripts/Folder Action Scripts/<launcher action scripts>.*

This program does significant checks and repairs on startup or at user request and often fixes problems, but occasionally the Finder and/or System Events is fussy and likes to deny it, so if you have problems, especially on install, quit ***Micalin's Launcher***, restart it, and try again.

Please let me know about any problems. If you have an issue with your system you think is from this software I can likely help you solve it so *DON"T DELETE YOUR HARD DRIVE BEFORE ASKING FOR HELP.*

**Visit the Micalin's Launcher Homepage for the most up-to-date information.**

## *Overview and Layout of Micalin's Launcher*

***Micalin's Launcher's*** functionality can be broken down into three basic overlapping activities around which its interface is designed.

1) **Organizing and customizing the Launcher (Dock) ƒ.**

2) **Installing, linking, updating, and customizing its folder actions.**

3) **Editing, customizing, and managing its filters.**

The user interface in turn divides these functions among five core windows and three core menus that reflect these basic functional divisions. In general the interface is organized around selections made the the Launcher Outline View window which shows the structure and content of the Launcher (Dock) ƒ and its subfolders.

## *A Word About this Users Guide*

This Users Guide might seem surprisingly long for a small bit of freeware...well, damned but IT IS LONG. As an excuse let me explain that its authoring was part of my development process to help me think through the interface and this verbosity is the result. It hopefully will server as a good references if needed.

The QUICKSTART document assumes most functionality is obvious from the interface and is all most people need to ever read.

***This 2.0 Guide is being updated based on the 1.1.2 Guide. However while Micalin's Launcher is in beta the User's Guide is likely to be incomplete or with some outdated information. It is beta as well.***

# 2. Requirements, Installation and Upgrading

## *Technical Requirements*

Any Macintosh running Panther 10.3.x with the BSD subsystem installed (installed by default on most systems). This version can also be used in 10.4 Tiger but testing has been limited. Version 2.x has also been specifically modified to be reasonably friendly to 800x600 displays.

This was originally written on Jaguar but it *might* require Panther for aspects of AppleScript Studio unique to newer OS X versions. I have NOT tested it on Jaguar and I do not know if key changes in AppleScript follow the compiled application as part of its dictionary, or follow the Macintosh Operating System as part of its AppleScript runtime environment. AppleScript Studio is so odd in this regard it is hard to predict. If it works on your machine, let me know.

## *Installation*

After an Introductory dialog repeating (with brevity) the Introduction from this Guide, you are given options for *Install* and *Install Without Scanning,* from which you normally always choose *Install*.

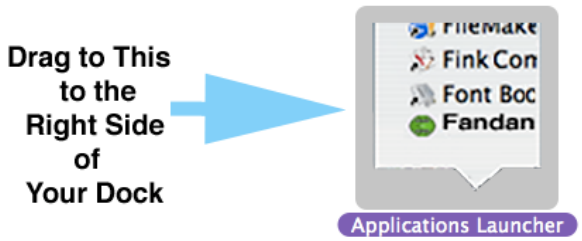
**Install**

Creates the directory structure, installs and activates the folder actions, and populates your Launcher folder with aliases from the applications in your Applications folder and it's Utilities folder along with linking the source folders to their subfolders for Auto-Adding alias to new applications. In addition the user can choose to scan and create linked subfolders for the Applications (Mac OS 9), Developer and the Applications folder in your home folder, if present.

**Install without scanning**

Creates the directory structure and sets up the folder actions but doesn't populate the Launcher folder with any aliases. It is for situations when someone wants to add a completely customized Launcher and not wait through the scanning of the Applications folder, or when someone already has a Launcher *ƒ* they want to use but need the folder actions and/or directory structure installed and setup.

After installation you will be presented with a Confirmation window with statistics on what what done during installation, some explanation and tips and a button to *Reveal (the) Launcher ƒ* in the Finder. Clicking this button shows you the populated Launcher (Dock) folder that you can then drag to the right side of your Dock.




*Anytime Micalin's Launcher starts and finds the Launcher (Dock) folder missing it will redisplay its Welcome and Install screens and prompt you to re-install. Preferences and filters are preserved if present.*

## *Upgrading (Normal)*

***Micalin's Launcher*** stores its version number in its preferences file so that whenever a newer version runs it knows to "upgrade". Upgrading updates (recompiles and replaces) all the folder actions (the Auto-Add actions and the helper action) with new versions, and forces a full repair of the Launcher (Dock) folder, all its folder action links settings, and the Launcher's preferences.

## *Upgrading from a pre 2.0 version to a 2.x version (Extended)*

For reasons I can only ascribe to self-hate or sadomasochism I changed a bunch of things about ***Micalin's Launcher*** I had intended to be fixed and stable therefore requiring me to address such when upgrading from pre-2.0 versions. Bad Rick..bad...

Specific changes that need to be addressed include the following:

1. **Creating custom folder actions plus reference entries/files for all existing Launcher subfolders**
2. **Linking the Utilities subfolder to its creation source folder (if the Utilities subfolder exists and is recognizable)**
3. **Allowing an upgrading user to create links to the same folders as if a new install**
4. **The name of the two original folder actions have been changed from 1.x to 2.x**
5. **The icons for the Dock folder and the Launcher subfolders have been changed from 1.x to 2.x. (yea, I know they were ugly...I was rushed)**

Most of this functionality is present but not completely debugged. However as of this writing (reflecting 2.0 beta build 3363 and 3364) the code to update the icons has not been implemented (since I just made new icons). The icons can be easily manually updated and any new subfolders or a subsequent re-build will use the correct icons. The new icons are included in a separate folder with the beta release.

If upgrading (rather than re-building) is not necessary, especially if the structure of an existing Launcher folder is basically unmodified, then reinstalling is probably preferable as it avoids any existing upgrade issues and creates new folders with the new icons. Re-installing can be accomplished either by deleting the existing Launcher (Dock) folder plus any folder actions of the form *"ML somename FA",* or by selecting "Rebuild Launcher" under the Launcher (Dock) *ƒ* menu but essentially does the same thing along with some preparation and cleanup. The Dock icon will update after a time (its uses a cache) but you can update it immediately by dragging the old icon out the Dock and the new revealed one into the Dock to replace it.

**Subsequent upgrades will be the limited *Normal* upgrade described at the start of this section and should be free of these issues.**

# 3. The Default Window “Main” and (Application Preferences)

The Main window is normal startup window and the Application Preferences window (also linked from the Preferences menu item) allows changing of settings that apply to the Micalin’s Launcher application independent of the folder action settings.

## *Main Window*

The Main window is the initial presenting screen on a normal run or after install or upgrade completes and provides a brief overview plus instructions, feedback on whether settings errors were discovered on startup or repair, and interface controls linking to windows for setting options and/or customizing the Launcher.

**Preferences**

Set preferences and options for the main application including activating/deactivating run and error logging. Along with controls to Reset All and Uninstall *(Reset and Uninstall or not working yet)*.

**Action Settings**

Set preferences and options for the Auto-Add and Helper folder actions including activating/deactivating run and error logging (applies to all actions), activate/deactivate the helper or all auto-add actions, or change settings for verbal or dialog notification on when adding. You can also change Helper actions functions including Simulate Application which closes the Launcher (Dock) folder and launcher Micalin’s Launcher whenever the Dock folder is opened, plus other related choices.

**Filter Settings**

Builds and opens a window with three table views representing the three filter sets allowing you do activate/deactivate any specific filter along with options to add or delete filters.

**Launcher Outline**

Builds and opens a window containing an outline view of the Launcher $f$, but with the application alias files in the main Launcher $f$ represented in a subfolder-construct titled “Laucher folder (main)” that is manipulated like another subfolder except it cannot be deleted or renamed. From this window you can delete or rename any alias or subfolder or perform various menu driven functions by selecting aliases and/or subfolders. Selections can be discontinuous.

**Show Repair Report**

Opens a window giving details for the most recent repairs.

This button is only active is the startup self-diagnostics found (and presumable repaired) a problem or if a Full repair was ordered for any reason. When active red text to the right will also alert you to the fact that a repair was performed.

**User’s Guide**

Opens and HTML version of this guide in your default web browser.

# *Preferences Window (for the Micalin's Launcher application)*

This window sets preferences that apply to the main Micalin's Launcher application (as opposed the the folder actions) and also contains Reset and Uninstall options.

**Enable Application Error Dialogs & Logging**

Turns on logging of errors for the main Micalin's Launcher application. If this is turned off most many dialogs are also disabled. If error logging is OFF any errors messages are directed to stout and will be visible in the console. Its default is ON.

**Enable Actions's Detailed Run Logging**

Turns on run-time logging for the main Micalin's Launcher Application. Logs are extensive and probably should not be turned on unless you are trying to fix a problem, or just curious. These are mainly used for debugging and development. Its default is OFF.

**Show Repair Report**

Opens a window giving details for the most recent repairs.

This button is identical to the button in the Main window but differs from that button it is always active here.

**Reset All** ***Not Yet Available***

Deletes the Launcher folder, all folder action scripts, and all preferences and filter settings and performs a virgin install. Similar to the Re-Build Launcher option except that option preserves the filters and preferences.

**Uninstall** ***Not Yet Available***

Give options for a complete uninstall, or a partial uninstall that leaves the Launcher (Dock) folder and all its subfolders intact but removes the Micalin's Launcher application, all its folder actions links and the folder actions scripts.

# 4. Managing & Customizing the Launcher ƒ

This section describes how to use the ***Micalin's Launcher*** interface to modify the Launcher (Dock) ƒ. Most functions involved the Outline View window.

> *The Applications Launcher ƒ really is <u>just a folder</u> that you can modify directly using the Finder. The interface serve several purposes, such as enforcing the consistency and formatting of Launcher ƒ items, allowing capturing deleted application aliases so as to add them to the filters, and tracking of auto-add links. Also the interface generally seems quicker and easier than the Finder when used for its limited role. Many of these functions would of course be easy to do by just opening the Launcher folder in the Finder and manually creating subfolders, copying icons, etc. However, In order for the Launcher to "look good" its subfolders need to have correctly structured names the form of which took some time to work out and it's easy to forget the that exact form or be tempted to waste two hours struggling with precisely how characters sort in list views. More dramatically, moving or deleting large groups of aliases from an open Finder window is the computer equivalent of the Limbo. No sooner do you have about fifty aliases selected then you accidentally double-click the next one and watch while fifty plus applications open at once! I hate that...*

**IMPORTANT:** In General changes cannot be undone. For example, once subfolders or alias are deleted the deletions cannot be reversed.

## *Methods and Options for Adding Aliases for New or Additional Applications to the Launcher ƒ*

Many functions related to adding aliases exist as part of this program but are so much more structural and intrinsic to its function when compared with operations such as "Create New Subfolder..." that its scanning and aliasing options might not be as clear from the controls at first glance. Therefore I think is worthwhile to briefly list the options for adding new aliases, the mechanics of which will be expanded on somewhat in the formal entries throughout the rest of this Guide.

- **Place files & folders into the Applications ƒ nor any folder linked to via and auto-add folder action**
- **Drop files & folders directly into the Launcher (Dock) ƒ where they are handled by the folder actions**
- **Drag and drop files & folders onto the Micalin's Launcher application icon to scan and alias them into the root Launcher folder**
- **Open files & folders to scan and alias using the File Menu**
- **Use the Launcher ƒ menu to scan a target folder as a source for a new subfolder**

## *The Launcher ƒ Outline Window*

Management and customization of the Launcher (Dock) *ƒ* is designed around its Outline View. This view is built from the Launcher *ƒ* the first time the Launcher *ƒ* window is displayed on every run of Micalin's Launcher. It mostly works like you would expect with some caveats relating to either my stamina, programming inadequacies, or other limitations related to AppleScript Studio.

With the exception of the *Applications Launcher ƒ* construct any subfolder or alias can be renamed, deleted individually or as part of a multiple selection, or included in a consolidation operation to from which to create a new subfolder. Application aliases may be copied from the Launcher *ƒ* or any subfolder to any other location in the Applications Launcher *ƒ*.

Once built the Outline View is either updated directly to reflect each change or rebuild if the changes are very complex. Errors or bugs could result in the Outline View diverging from the actual Launcher *ƒ* but the outline view can be rebuilt from the physical Launcher folder at any time by clicking the *Refresh Outline* button in the *Outline View* window.

Micalin's Launcher version 2.0 *should* now rebuild the Outline View in all cases where such action is always functionally required, such as after running filters manually.

This window also contains the toggle setting to "Automatically Add Deleted Applications to Excludes Filters". When **ON** this setting causes any application aliases the user selectively removes, whether individually or as part of a group selection, to be added to the Excludes filters to prevent that application from being re-aliased in the future. Applications not specifically selected but deleted as a consequence of a parent subfolders deletion are not added automatically. This setting is **ON** by default.

## *File Menu and Related Options*

Since this is not a document-based application the file menu is relatively unused. Currently its only functioning option is to manually choose files and/or folders to scan for applications to alias.

**Scan Files/Folders ...**

This is technically similar to dropping files and folders on the Micalin's Launcher application but differers in that the Launcher folder in to which alias are placed can be selected. File and folders dropped directly onto the Micalin's Launcher application or files and folders opened with the menu option are scanned and aliased just as if they had been added to the applications folder except any resulting aliases are placed into the Launcher folder/subfolder selected in the Outline View. If the Outline View is not open or no Launcher subfolders are selected then the aliases are placed in the root Launcher folder, Applications Launcher (main). The original chosen files and/or folders are not moved or modified.

# *Launcher (Dock) ƒ Menu*

**Show Launcher Outline window**

Opens the Launcher Outline View window including building the Outline View if opened for the first time on each run of Micalin's Launcher.

**New Empty Subfolder ...**

Creates a new subfolder in the Launcher *ƒ* with a user supplied name.

The subfolder is given the custom subfolder icon and the requested name is validated, checked for duplication, and coerced to correct form for a subfolder.

**New Subfolder from -> Consolidation of Selected Items**

Takes the set of subfolders and links (excluding the main Launcher *ƒ*) plus any aliases selected in the Outline View and combines then into a single new subfolder with a user-supplied name.

If the supplied name matches an existing subfolder then the user is given the option of adding the selected contents to that subfolder. The selections can be discontinuous and across all subfolder and the main Launcher *ƒ*. Alias selected residing in any subfolder are removed from that subfolder and placed in a new subfolder. Any subfolder selected is deleted and all its contents placed in the new subfolder. If part of the original selection included a subset of files from a subfolder that was also selected then all the aliases in that subfolder are selected since the selection of the parent subfolder takes precedence. Any auto-add links to the selected subfolders are transfered to the new subfolder and the original selected subfolders listed in the new subfolders get info window..

**New Subfolder from -> Chosen Folder as Linking Source**

Lets you choose a folder to use a source from which to create a linked launcher subfolder from.

The subfolder gets the source folders name and is populated with aliases to any applications contained in the source folder plus is auto-add linked to the original source folder. If the selected subfolders name matches an existing Launcher subfolder the user is given the option of adding the contents and making a link to that existing subfolder. This option contrasts with the Choose File/Folder to Scan option in the File menu that lets you choose a folder (or folders) to scan with those aliases going into the selected subfolder or main launcher folder without creating a new subfolder are creating a link. The source folder is not moved or modified.

**For example:** If the Developer folder was chosen in this option you would get a new subfolder named ' - Developer' containing aliases to every application in the Developer folder and with a new Auto-Add folder action linking the Developers folder to the ' - Developer" Launcher subfolder.

**Move Alias to -> <list of Launcher subfolders plus main folder as a submenu>**

Provided ONLY aliases files are selected (multiple and discontinuous selections are OK) in the Outline View this menu item provides a submenu of all the subfolders plus Applications Launcher *ƒ* (main) as destinations for moving those aliases. This submenu is updated as the Launcher *ƒ* changes.

**Get Subfolder Info**

*See the section on the Subfolder Information window at the end of this chapter.*

**Delete**

All selected applications aliases and subfolders selected in the Outline View are deleted (except for the Applications Launcher *ƒ*). Any subfolder selected is deleted along with all its contents even if part of the original selection also included a subset of files from that subfolder in addition to the subfolder itself.

**Apply Current Filters to -> All Launcher folders, or Selected Launcher folders**

This option lets you apply the mask and the excluded application filters to the Launcher folder or any of its subfolders either individually or the entire Launcher at once.

This is intended to remove various aliases that probably shouldn't be in a Launcher but do reflect actual applications, such as Electronic Registrations, Uninstallers, etc. These filters normally run each time new aliases are made. Note that this option runs the excludes applications filters as mask filters meaning. Normally these filters are only applied as aliases are created but never applied to the subfolders and only filter for exact matches but when selected manually exhibiit the behavior of the mask filters.

*See the section on Filters in Other Details for more details on the Mask and Excludes filters.*

**Clean All (Remove Broken Aliases)**

The option removes all broken aliases from the Launcher *ƒ* and its subfolders as a group. It cannot be applied individually and is not modified by selections in the Outline View.

Broken aliases are those that have lost their connection to their target file such that they no longer point to anything. This usually results from the original file getting deleted. If cleaning has any problems it reports an error count otherwise it simply reports the number of broken aliases removed. If errors are reported you should probably run Disk Utility to repair your drive.

*I am planning on expanding this option to allow the user to "Clean" any drive folder or the entire drive but I have to MAKE SURE it will not do damage first.*

**Show Launcher *ƒ* in the Finder**

This reveals the main Launcher folder (the Launcher (Dock) *ƒ*) which contains on the aliases and Launcher subfolders in the Finder. This option is to allow the user to add the Launcher (Dock) *ƒ* to the Dock it is missing.

**Rebuild Launcher**

Deletes the Launcher (Dock) *ƒ* and all the folder actions and rebuilds a new Launcher (Dock) folder is if a fresh install including using the New Install dialog.

The preferences including any filters and those filters are used for the rebuild.

***Rename Alias or Subfolder is an implicit function.***

Double clicking on any Outline View item title other than 'Applications folder (main)' will allow you to edit that title. Always conclude editing by pressing <*Enter*> or <*Return*> or the change will will not register. Renaming the 'Applications folder (main)' to anything or any other item to an existing or illegal name results in the item reverting back to its pre-edit name. Subfolders names are adjusted to their standard format of ' - *somename*' so entering the leading ' - ' is not necessary.

## *Subfolder Information Window (the Get Subfolder Info option)*

This window is called by selecting a single Launcher subfolder in the Outline View and selecting the *Get Subfolder Info* menu item, or clicking *command-i*. It displays the selected Launcher subfolder's (or the main Launcher folder) source folder used for its creation (if present) or the Launcher subfolders combined to create it, the name of its Auto-Add folder action, and a list of all its linking folders. It has options to disable/enable any link, add/remove a linking folder, or reveal a linking folder in the Finder. This window generally updates dynamically with changes, or when a new subfolder is selected. The link count is the total number of links (active or inactive) linking to the selected subfolder.

You can a select linking folder from the list or deleting or revealing in the Finder. The selected folder has black text and a black outline.

**Active (the checkbox in the links list)**

Toggles whether the related Auto-Add link is active.

This is different than deleting a link If deleted no reference to that link remains but if disable in the Subfolder Information the link still exists but is turned of in the systems folder actions table. This is the only place where an individual link can be enabled or disabled.

*The Launcher was originally designed to only handle a singe Auto-Add link, to the root Applications folder, and a lot of code, especially in diagnostics and repair assumed that. It was only late in the development that I decided to allow the user to inactivate this core link so some bugs may persist when doing so.*

**Add New Linking Folder**

Using a standard file dialog allows the user to choose an additional drive folder to Auto-Add link to the Launcher subfolder.

**Delete**

Deletes the Auto-Add links (as opposed to making it inactive) for the linking folder selected in the Subfolder Information window.

**Reveal in Finder**

Shows the actual linking folder in the Finder for the linking folder reference selected in the Subfolder Information window. Displays an error if the folder does not exist and calls for a repair.

# 5. Managing Folder Actions

This section describes the settings and options for the Auto-Add and Helper folder actions. Many of the options under the Folder Actions menu are linked to selections in the Outline View window and can be considered a continuation of options to modify the Launcher, specifically its behavior when adding application aliases automatically.

## *Micalin's Launcher's Folder Actions*

Micalin's Launcher installs two folder actions types, the Auto-Add actions and the Helper action.

The Auto-Add actions watch drive folders and scan any files or folders added to them for applications that if found are aliased into the linked Launcher folder. Every Launcher folder (subfolder) including the root Launcher folder has a single unique Auto-Add action assigned to it. Each action, or by extension Launcher folder, can link from any number of drive folders but each drive folder can link to only one Launcher folder (i.e, a drive folder cannot have more than one Auto-Add action attached to it).

These Auto-Add actions use the filters as set at the time they were created or updated (compiled). They use the excluded applications filter while making aliases and run the masks filters after aliasing is complete. These are large scripts that and a large percentage of its code with the Micalin's Launcher application.

An Auto-Add action is also linked to its parent Launcher subfolder at compile time and automatically disables itself if it cannot find that parent Launcher folder when it runs. This can be repaired by updating a Launcher folders Auto-Add action using the Folder Actions menu.

The Helper actions role is comparatively simple. Its primary job is to transfer files or folders dropped onto the Launcher (Dock) icon (and thus into the Launcher (Dock) folder) to the main Applications folder providing a simple method of adding Applications when the Applications folder is not readily visible. It has acquired some other odd jobs as well including: It briefly scans added items to see if they are composed of only aliases or folders of only aliases and, if so, keeps them in the Launcher folder. It can either display a reminder that it is disabled when the Launcher ƒ is open with the option to open the Micalin's Launcher application, or it can close the Launcher ƒ immediately when that folder is opened and open the Micalin's Launcher application instead. The Helper action is always disabled when the Outline View is open.

Most options can be disabled via the Actions Settings preferences window.

*Note: The easiest way to manage a Launcher folder's Auto-Add links is to use the Get Subfolder Info menu item to open the Subfolder Information window. This is also the only way to "disable" individual Auto-Add links without deleting them.*

## *Auto-Add and Helper Action Settings Window*

The preferences window contains the toggle settings (ON/OFF) that control the behavior of the Launchers folder actions. Most preferences are read each time an action runs, unlike the filters and most of the key reference variables which get compiled into the folder actions as they are created. Any preferences which have a corollary in the main applications preferences apply only to the folder actions.

**Enable Auto-Add Actions**

Toggles whether the Auto-Add actions are active as a group.

This setting applies to ALL Auto-Add actions at once and changes dynamically as the checkbox is clicked. It matches the related menu item. This settings can be used to disable the Auto-Add behavior of the Launcher if the user wished to only add items manually but continue to use the Launcher. Its default is ON.

**Talking**

Toggles whether verbal feedback is provided by the Auto-Add actions when scanning new files and folders and for any application aliases added or updated. Its default is OFF.

**Show Dialogs**

Toggles whether dialog feedback is provided by the Auto-Add actions when scanning new files and folders and for any application aliases added or updated.

The dialog is presented by the Finder and automatically dismissed after a few seconds. The dialog generally consists of the number of files/folders scanned and the number of new application aliases added., Its default is OFF.

**Enable Helper Action**

Toggles whether the Helper action is active.

When OFF none of the Helper action's functions are active and the Helper action does not run or monitor at all. This is different from its active but "disabled" state present when the Launcher (Dock) folder is open. Its default is ON.

**Display Disabled Reminder**

Has the Launcher folder display an auto-dismissing reminder when the Launcher (Dock) folder is opened (usually by clicking its Dock icon) stating that its controlling folder action is inactive while the folder is open and giving the option of closing the Launcher (Dock) folder and opening the Micalin's Launcher application. This setting is cannot be set to ON as the same time as the settings *Simulate Application* or *Re-Close the Launcher (Dock) ƒ*. Its default is OFF (as required by *Simulate Application* being On).

**Re-Close the Launcher (Dock) *ƒ***

This settings causes the Launcher *ƒ* to immediately close when that folder is opened (but not necessary open the main application). Its default is ON (as required by *Simulate Application* being On).

**Simulate Application**

This settings causes the Launcher *ƒ* to immediately close the Laucher (Dock) folder when that folder opens and open the Micalin's Launcher application instead, *supposedly,* making the Launcher (Dock) icon simulate a running application icon.

The Re-Close the Launcher (Dock) *ƒ* settings is automatically enabled when this setting is active. Its default is ON.

**Enable Actions Error Dialogs & Logging**

Turns on logging of errors for the folder actions. If this is turned off most error dialogs are also disabled however as of version 2.0 most error dialogs are disabled regardless of settings. If error logging is OFF any errors messages are directed to stout and will be visible in the console. Its default is ON.

**Enable Actions's Detailed Run Logging**

Turns on run-time logging of all folder actions. Logs are extensive and probably should not be turned on unless you are trying to fix a problem, or just curious. These are mainly used for debugging and development. Its default is OFF.

# *Folder Actions Menu*

The Folder Actions Menu allows for enabling or disabling the folder actions, adding or removing Auto-Add links, or updating (recompiling) a selected Launcher (sub)folder Auto-Add actions or ALL folder actions. It also contains options for manually running the self-repair routines as most of its repairs are related to the folder actions or links.

**Show Action Settings window**

Opens the *Auto-Add and Helper Actions Settings* window. **Auto-Add for Applications Folder**

Equivalent to the Preferences check box this toggles the functionality of having the Applications folder watched for added applications in order to aliases them. Has a ✔ by it if active. The Preferences window has a bug and it does not set the state of this menu item when you change the Auto Add setting when it changes it but the check box is always correct.

**Auto-Add Linking Actions Enabled** and **Launcher *ƒ* Helper Action Enabled**

Toggles the global enabled state of either all the Auto-Add actions or the Helper action and is equivalent to the check boxes in the *Auto-Add and Helper Actions Settings* window. There is a check by these menu options when enabled.

**Create Auto-Add Link from New Folder**

Brings up a standard file dialog for choosing a folder to attach to the Launcher folder selected in the Outline View.

**Remove Auto-Add Link -> <submenu of list of links of the form 'drive folder => launcher folder'**

This option shows a submenu of ALL links, both active and inactive. Inactive links have the '=>' replaced with a '-|-'. When you choose an Auto-Add link to remove all references to that link are deleted.

*Note that this is a somewhat redundant option and really replaced by the Get Info option and thus has some gaps in its implementation. For example, when a link is disabled using the Get Info window its listing in this submenu will not change until Micalin's Launcher is restarted*.

**Update Subfolder's Auto-Add Action**

Recompiles and saves the selected Launcher folders Auto-Add folder action updating its references and filters.

This is the method to update an Auto-Actions filters or repair an malfunctioning or non-functioning action or an action that disabled itself after losing its target Launcher folder.

**Update All Actions (Helper and Auto-Add)**

Recompiles and saves new copies of ALL the Launcher folder actions, including all Auto-Add folder actions and the Launcher folder Helper action.

This is always done at each upgrade and if done manually is non-destructive and only takes as few seconds.

***I**nterrogate Options* ***Not Yet Available***

These options are not yet active but will allow you to view an subfolders statistics and its Auto-Add action scripts current compiled in filters.

*The folder actions can be interrogated and they do output most information but the display of it is not finished.*

**Full Verify and Self-Repair**

Orders a Launcher Full self-repair. A full repair is always run after an upgrade or for certain error conditions and can be called by any Launcher subroutine or folder action (folder actions do not currently call for repairs.)

A Full repair attempts to mesh the physical launcher folder, Micalin's Launcher's links and structure databases, and the actual folder action settings for the operating system. A Full repair also attempts to identify any lost Launcher folders and incorporates any foreign folders it finds in the Launcher folder assigning them a custom icon and compiling a custom Auto-Add action. Foreign folders it finds that contains actual files (not aliases) or nested folders it assumes are a result of a failure of the Helper action and so relocates these folders to the Desktop to a folder titled "Moved from Launcher" in a subfolder with the date and a time index (for example *~/Desktop/Moved from Launcher/March 30 (time index 14870)/<moved folders>* ).

A full self-repair is also called when Micalin's Launcher preferences, which contain its databases, are found deleted. A Full repair can rebuild those databases including Auto-Add links from the system settings and physical Launcher folder.

A brief self-diagnostic is performed each time Micalin's Launcher (the application) is run and if errors are found a much more constricted *Limted* repair is ordered. A *Limted* repair mostly resets the Auto-Add action assigned to the main Launcher folder along with its Helper action, along with some other maintenance and repairs on the folder action settings, and is an updated an expanded version of the self-repair performed in the 1.x versions.

Both a *Limted* and *Full* repair look for and remove ANY folder action for which the actual drive folder referenced by the folder actin has been deleted or is missing. *This is called deleting orphans.*

**Show Launcher Repair Report**

Opens a window giving details for the most recent repairs, or the startup check if that is most recent. This window can also be displayed from the main window if it contains anything interesting.

*This window is generally of NO INTEREST to anyone but me but does include a lot of information in the event of major repairs.*

## *Startup Report Window*

This window provides some information on what the Launcher found wrong and what it did about it when it was starting up for its current run or if a repair was ordered. Each time the Micalin's Launcher applications is run it performs some brief checks and, if necessary, repairs to key settings and/or replaces missing components. Relatively thorough checks are performed if a Full repair is ordered. This window is of little importance to most users and some display glitches remain.

## *Rules for Scanning and Moving Files and Folders*

Except for its Preferences file, logs (if active), the Launcher folder structure at *~/Library/Application Support/Application Launcher Data/<Application Launcher/*( all aliases & subfolders), and the adding of folder action scripts to *~/Library/Scripts/Folder Action Scripts* no files or folders outside the Launcher folder s are ever modified or moved except for two cases. These are: 1) modifications to the Applications folder when items are moved into it from the Launcher folder, but that action does not replace files without permission, 2) modifications to the users Desktop when files are relocated to it during a Full repair. This is to make a sure as possible the Launcher does not do damage or delete applications. Files and folders (supposedly just folders and aliases) in the Launcher folder are fair game for this program to modify as it needs.

### *How aliases added to the watched folders are handled.*

File aliases (what the Launcher makes and manages) when not part of mixed group are treated specially. It is assumed that if you are dropping collections of pure aliases (free or contained in folders) into the Launcher (Dock) folder you are doing so intending to add to the Launcher's collection. Therefore such collections are left alone and not moved to the Applications folder. If contained within a folder they are left in that folder effectively providing another menu. The added folders name is changed to the standard launcher form and a custom icon assigned with the next full repair and by *2.0 Final* will be changed by the Helper action which will also order a full repair. Normal folders with mixes that include aliases are moved intact to the Applications folder as described above.

### *The Rules for Scanning and Adding Application Aliases*

- Every time items added to the applications folder are scanned for aliases a finishing filter is run at completion to remove likely low use aliases. It is applied to the the target launcher folder at its root level only. The Excludes filter is run while aliases are actually being created therefore never applied (unless run manually) to any existing aliases.

- Dropping files/folders onto the Launcher Dock folder is the same as placing them into the Applications folder in that those files/folders will be immediately moved to the Applications folder and then scanned. Many users confuse the Launcher (Dock) folder with the Applications folder often dropping new applications into it assuming that they will be added to their applications folder. This seems reasonable so the Helper folder action just moves those items to Applications as their likely actual intended destination. Sometimes it really is just easier to get to the *always-visible* Launcher (Dock) folder.

- Aliases in isolation when added to the Applications folder get mirrored in the Launcher Dock folder (copies are placed into the Launcher folder but the originals are left in Applications). Aliases in isolation added to the Launcher (Dock) folder are presumed to be for the Launcher and are not moved to the Applications folder but

remain in the Launcher Dock folder. A folder containing only aliases is the same and remains in the Launcher Dock folder as a folder that can provide nesting of pop-up menus. *Note that it will eventually be incorporated by self-repair.*

You can customize your Launcher's Dock folder by manually rearranging it's content. Doing this requires that the Simulate Application option to be turned OFF so the folder will stay open. In that case the watching action for the Dock folder is disabled while the folder is opened so that it doesn't disturb the house cleaning by actions such as immediately moving away any new folders you create to the Applications folder. The folder action reactivates when the Dock folder is closed. The folder action is only aware of activity while it was active so nothing done while it was inactive will be changed.

*Note: The folder action is not actually disabled, just aware that the Launcher folder is open and therefore it bypasses all functions until it is closed. If it were actually disabled it would not be able reactivate when the folder was closed since it no one would be watching.*

## *The Micalin's Launcher Application*

The code for scanning files and folders and adding aliases to the Launcher is identical in both the Micalin’s Launcher application and the Auto-Add actions. Also, like the those actions, every time new files or folders are scanned for aliases a the Excludes filters are run during scanning and the Mask filters at completion. Except for varying to Launcher folder into which aliases are added dropping files/folders onto the Micalin’s Launcher application, opening them with the Scan Files/Folders ... menu option, or dropping them on the Launcher (Dock) folder icon is equivalent in outcome to dropping them into the directly into the main Applications folder.

The Micalin’s Launcher application of course has a number of manual scanning options not appropriate to the Auto-Add actions.

# 6. Managing Mask and Exclude Filters

This section describes how the filters work, and how to customize, manage and remove them. Much of the filters functionality is automatic as it tracks which alias files your delete so the launcher will likely work well even if filter settings are never manually adjusted.

## *Mask Filters*

Mask filters are text strings representing key portions of file names and match to a file if the file's name contains the mask. They can be used to filter a large number of aliases with a single mask by choosing a common name element, such as "Editor" or "Microsoft", or made very specific by using a long text phrase likely to only match a single specific file, such as "Photoshop Elements 2.0". These filters are used by both the main application and the individual Auto-Add folder actiosn after a set of files and folders have been scanned and aliased and are applied to the folder receiving the aliases, usually the Launcher (Dock) *ƒ*. This means in generally any linked to Launcher subfolder will be filtered occasionally, and some, like the Launcher Folder (main) are likely to have mask filters applied often. If you wish some alias to remain in such a folder either rename it to avoid the mask filters or disable or delete the offending mask.

There are two groups of mask filters. The first is the Defaults Masks and are the ones that filter on a new install. They can be inactivated but not deleted or modified. The second group is User Defined Masks where new masks can be added, edited and deleted.

## *Exclude Filters*

Exclude applications are applications names used by the main application and the Auto-Add folder action swhile aliases are being created to exclude aliases for files with names exactly matching any one of the exclusions. New exclusions can be added by choosing applications via a file browse dialog. They can also be added automatically by choosing the option to "Automatically Add Deleted Applications to Excludes Filters" in the Outline View window. Exclusions can be deactivated and deleted but they cannot be edited.

Since the Excludes filters are used during aliasing to exclude matching applications before they are aliased, and not applied directly to the target Launcher folder the Excludes filters *do not* effect the existing contents of Launcher folder. The exception is if they are applied manually from the Launcher (Dock) *ƒ* menu where they are applied *as mask filters*.

# *Filters Settings Window and the Filters Menu*

The filters window has a table views of all the filters and works in concert with the Filters menu to edit, activate, manage, and delete the mask and exclude filters.

**Show Filters Settings window.**

Opens (building if necessary) the filter settings window with its table views.

**Add New Empty Mask**

Makes a new disabled Mask item in the User Defined Masks table with the default string "–Double Click to Edit and Activate New Mask—". Once this text is modified it cannot be reverted back to that string and if a filter with that string is activated it is excluded from the final deployed filter. To edit and activate this new masks double click its text edit the text string. Masks activate automatically when edited but can be again deactivated. Do not use wild cards on the outsides of the string as those are applied by the filters. A standard Unix wild card should work when used on the inside of the mask's text string.

**Add New Exclusions**

This allows the user to select one or more applications using a standard application file-browse dialog or optionally a standard file-browse dialog. The visible names of any applications selected are added to the filter list and activated.

**Delete**

Delete User Defined Masks or Excluded Applications that the user has selected in the table views. Selections can be multiple and discontinuous. Default Masks cannot be deleted.

**Make Active** and **Make Inactive**

Activate or inactivate all selected Default Masks, User Defined Masks, and Excluded Applications.

***Editing a User Defined Mask***

Double clicking the text for a User Defined Mask will allow you to edit that title. Always conclude editing by pressing <*Enter*> or <*Return*> or the change will will not register.

**Reset to Defaults (button)** ***Not Yet Available***

Resets all filters to there default state. This includes activating all Default Masks and deleting all User Defined Masks and Excludes Filters.

# 7. Other Details

## *Multiple Users*

This application and its folder actions run in user space and its various specific folders and subfolders are kept in the user's Library allowing multiple user accounts to use individual Launcher folders on the same computer with individual and personal settings for each user. Like all OS X applications the main application does not need to be duplicated for multiple users but can serve multiple users from a single application, usually placed in the Applications folder.

## *Notable Limitations*

Applications removed from your Applications folder to do not have their Launcher folder aliases deleted, nor do applications removed from the computer. Attempting to do so would be intrusive at best and a disaster at worst. The Clean Alias function will remove broken aliases left from deleted applications from the Launcher.

Limited data is tracked regarding the existing folder actions such that if a folder action is removed completely from the folder actions table by removing them using the ***Folder Actions Setup Utility*** then ***Micalin's Launcher*** does not try to reconstruct them. Instead it alters its database to reflect the change. Therefore links in such a fashion would have to be recreated if desired. Although the repair and diagnostics are complex, in general the actual state of the folder actions table is taken as the last word and the baseline for most repairs. The flip side of this is that ***Micalin's Launcher*** can completely reconstruct all links, menu items, and its data given an intact table folder actions table and usable Launcher folder to work from.

## *About the Folder Actions*

Folder actions, for those unfamiliar with them, are a standard feature of the Mac OS dating to before OS X. They are AppleScripts that are associated with folders such that they are activated for certain events involving that folder, such as the folder being opened, or having its contents change. Folder actions are called and run by the System Events application, a faceless background application that handles lots of functions related to scripting including such features as special needs UI interaction using Assist Devices, speakable items, UI scripting, etc. ***Micalin's Launcher's*** folder actions are standard AppleScripts run by this System Events that then manipulate and control the Finder and the System Events application, along with issuing shell scripts, to handle there functions. You can attach the Launcher's installed folder actions to other folders manually, in addition to the ones ***Micalin's Launcher*** attaches them to, and they should work fine. You can also manage folder actions via the *Folder Actions Setup Utility* located in your Applications folder (probably in the AppleScript folder.) *It will not be aliased in your Launcher folder since the filters catch it. It has Setup in its name. Code for avoiding such accidental filtering is in place but not finished.*

## *About the Filters*

There are lots of "applications" in your applications folder that don't belong in a pop-up Launcher but are technically applications. These are things like Registration applications, Installers and Un-installers, droplets, background and daemon applications, and many others ... Since it is a pain to pick them out of your Launcher folder I just filter them based on some common name elements. This really cuts back on the trash that lands in the Launcher folder and makes using it much more pleasant. These name elements are the masks filters and the name elements the masks

The current mask filters just remove everything from the target Launcher folder with names matching certain phrases (text masks).

Currently those are:

"update", "Update", "install", "Install", "About", "about", "Regist", "regist", "Setup", "setup", "demo", "Demo", "read", "Read", "assistant", "Assistant", "Tour", "tour", "Alias", "sample", "Remove", "remove", "Drop", "HP ", "exe", "Daemon", "daemon", "Microsoft Query", "Microsoft Clip Gallery", "Microsoft Error Reporting", "Microsoft Cert Manager", "Microsoft Database Daemon", "Microsoft Office Notifications", “Profile”, “profile”, “Welcome”, “Limewire at Login”

“HP ” is present and set to ON since I have dozens of HP related trash apps from my printer drivers but since HP is so generic I have concerns it might over match. This filters was disabled by default in the 1.1.x versions but is now on with the addition of a trailing space. Time will tell if it is too much. If you have any suggestions, especially for file groupings I don’t have but are likely to be on other systems please let me know.

Excluded applications are intact and complete names of applications to exclude from aliasing. Those names have to match perfectly and both the main application and the Auto-Add folders action apply these excludes before the alias is created. Any aliases the user deletes from the Launcher ƒ has its name (hopefully the same as its originals name) added to the excludes list.

This particular choice of filtering is performance based. I do not want the folder actions to use too much processor time when they are called. Since it is trivial to compare each application to a single long mask (of excludes) to see if that mask contains the application I apply that to each application found. This imposes almost no overhead while aliasing. It is much more time consuming to have to compare each application to see if their names contain any one 40 or more masks. Therefore the masks are implemented as a shell script (takes about a second to run) that is run just before the folder action quits.

## *About the Logs (Error and Run logs)*

The Micalin's Launcher application and all folder actions have the ability to write out logs, both run logs of what they did each time there were active, and error logs. I use the run logs for debugging and they are OFF by default in the Release version (or they should be). The logging is pretty extensive and carries significant overhead and unless you are curious or trying to solve a problem there is no reason to turn them on. If you do you can always turn them back off. Some run logging always occurs as the main Applications is started due to the fact that the Logging routine is a separate script that checks the preferences after is is loaded to see whether to log, which is after a short time after the main application is in initial startup and already logging. The Micalin's Launcher application and every folder action writes to its own run and error logs.

The error loggin for the main application and the folder actions defaults to ON since it does most of the error processing for any errors Micalin's Launcher cannot handle internally, generally errors that are unpredictable or that prevent to program from doing something essential to its function. It passes the system messages generated by those errors on to the user as REALLY OBSCURE error dialogs courtesy of AppleScript Studio's reliance on Cocoa for error feedback. Every time the *write_error_log()* routine gets called it beeps twice, writes to the error log, and then posts a error message in a special error window. Those error dialogs were in the Finder in the 1.x versions but should now generally have a dedicated error dialog window from the Launcher. The folder actions still must use the Finder. I made this change to address the annoying tendency for the Launcher to display an error dialog in the Finder unknown to the user, then give a series of error as the Finder timed out since the unanswered and hidden error dialog was holding the Finder hostage.

The 2.x version of Micalin's Launcher generally has ALL error dialogs (but not logging) disabled for the folder actions regardless of the settings. The code for those actions is very preserved and is it seldom a user wishes to be bothered by an folder action error.

If you are getting error dialogs (and beeps) but they don't seem important turning of the error logs also turns off most of these dialogs and beeps as well. Certain errors are logged or displayed regardless of settings if they are considered important enough.

The logs are kept in ~/Library/Logs/ and the filenames are as follows: The main application writes to "*Micalins Launcher (error or run.log)"*, the Auto-Add folder actions write to "*ML sf_<subfolder name> FA (error or run.log)"* and the main Launcher folder Auto-Add action to "*ML Launcher FA (error or run.log)"*. The Helper action writes to "*ML Helper FA (error or run.log)"*. Note that these names are changed from the 1.x version. For example the Helper action did write to "*ML Launcher FA (error or run.txt)*".

You can view the logs with the Console applications in your Utilities folder. Just click on Logs and flip down the triangle by ~/Library/Logs .

## *Do I have to use the Folder Actions to use the Launcher?*

No. If you think the folder actions are slowing your computer down (they shouldn't, they are only called if needed) or if for any reason you want to use the Launcher but not its folder actions you certainly can and it works like you would think. Just disable the folder actions but keep the Launcher (Dock) folder in the Dock. It is JUST a folder and sitting in your Dock it consumes no more processor power then any other folder in the Dock. You can still add aliases to it by dragging them onto, or opening them with the Micalin's Launcher application, plus use all its management tools. You will just not have automatic adding of aliases or relocation of items from Launcher folder to the Applications folder (although I would recommend leaving the Helper action on).

## *Uninstalling*

The code for uninstalling is in place but not yet active.

Do the following to get rid of each and every scrap or Micalin's Launcher. Personally I never throw away old preferences as I invariably end up reinstalling something again and those old preferences make it feel nice and at home.

1) **Use the Folder Action menu to disable ALL folder actions**

2) **Remove the ALL folder actions with names of the form *"ML somename FA"* from your user Folder Actions Scripts folder at ~/Library/Scripts/Folder Action Scripts/**

3) **Remove the *Micalin's Launcher* application itself.**

4) **Remove the *Applications Launcher Data* folder from your user Library at ~/Library/Application Support/**

5) **Remove the preference file, *com.nm.Launcher.plist*, from your user Preferences folder at ~/Library/Preferences/**

Although not critical since they are disabled, for completeness you might delete the Launcher's folder actions from the Folder Actions table using the *Folder Action Setup Utility* found in the *AppleScript* folder of your *Applications* folder. Delete all folder actions listed on the right that have either no attached scripts listed on the left, or only scripts of the form "ML somename FA". If a script with a name of a different form is present it does not belong to the Launcher. In that case delete all script from that folder action of the form "ML somename FA" but you might want to leave the folder action. If the folder action foreign script show disabled you may consider re-enabling them if you feel comfortable.

*It is somewhat unlikely anyone will have folder actions residing on there system they don't know about and understand.*

## *Version History, Ranting, and Other…*

For me this is version 3.3 of this application, so if you see any references to 3.1 3.2, or 3.3, this is it. 3.1->1.0, 3.2->1.1, and 3.3->2.0.

This has been a hobby project of mine for some time and so it has meandered and progressed for quit awhile with much of what I do with it experiments not related to its Launcher function. Given its true state and that I gave it only to friends until recently calling it 3.1- 3.3 would have been misleading. However considering how much this version has changed I just can't resist calling it 2.0. The original 3.1 version was released as 1.0.

### Version History (as of 5/05)

| | |
|---|---|
| Early | Ugly. You don't want to know. See the Technical Read Me if you must. |
| v0.8 | AppleScript application working as application and folder action ~ 2 years ago. |
| v2.0 – 2.1.1 X | AppleScript application as app and folder actions ~ 1 ½ years ago – This was a complete rewrite that produced most of the core code comprising the folder actions still used in 3.x. These version spun of the Helper action as a separate script. |
| v2.2 | AppleScript application as app and folder actions ~ 3 -4 months ago - fixed some annoyances, expanded rebuild code, and laid early groundwork for a UI expansion |
| v3.0 | Interval AppleScript application that wrote out both the Helper action and the original Applications Laucher (Auto-Add) action as separate scripts ~ 1/05 – This was a transitional version (never really used) to prepare for placement into an AppleScript Studio Xcode project. |
| v3.1 (1.0.x) | AppleScript Studio application release as v1.0 release ~ early 2/05 – added lots of long overdue functionality over 2.0 – 2.2 including filters (no user interface for them), menu and button driven Launcher *ƒ* management (no table or outline views), much better installers, better startup checking, and some performance tuning plus and AS interface to set options. |
| v3.2 (1.1.x) | AppleScript Studio application released as v1.1.x release ~ mid 2/05– added better Launcher *ƒ* management via the outline view, and filters management via the Filter Window, plus other features. GUI much more polished. |
| v3.3 (2.0.x) | Major upgrade including cleaning of most GUI and menu glitches, errors and annoyances, expansion of folder actions to allow linking from any Launcher folder to any number of drive folders with each Launcher subfolder having its own folder action, and Shell Script knights to recover folder actions that crash System Events, along with MANY other changes. Early TIger compatibility with first (plus a little) beta release. 3.3 only at beta as of this writing. |

## To Do List

Some for the 2.0 version, some for beyond (as of 5/05)

### beta 2.0 To Do List

- Modify helper action to integrate added alias only containing folders with custom icon and correct name format plus call for Full repair on the next run of the main application.
- Complete Interrogation function
- Complete uninstall and reset options
- Expand size of Get Info window (its role expanded beyond what I was thinking when I made it)
- Add several more internal help windows
- Quash many, many bugs

### 2.x To Do List

- Create a simple cron job for basic Launcher maintanence including cleaning of broken alaises.
- Options to select among various icon sets.

### 3.0 To Do List

- Change from folder actions (partialy or completely) to a objective C based routine.
- Move (perhaps optionally) the Launcher menu  to the menu bar (upper right).

## Who am I

I am clearly crazy.

## Naked Micalin Software?

It annoys my wife, and she is so cute when she is annoyed (well sometimes...)

It is also a really unique name, and sort of cool...in a way...

No, this is ABSOLUTELY NOT the leader for a porn site.  Also there is nothing for sale anywhere on the Naked Micalin Software site or the Sliced Apple site.

This is purely a hobby for me.

Oh, and my wife is insane...

# 8. Contact Information

If you have trouble, send me an email and I will see what I can do.

Rick Wikoff and Micalin Wikoff
rwikoff@mac.com or swevil@mac.com (for my wife)

Waste time on more freeware releases of dubious value at **SlicedApple (NMS)** or visit my much more interesting if obsessive website at **SlicedApple**.

***Come By and Say Hi!***