Best Practices in JMP® Add-In Management

WHITE PAPER

Table of Contents

Michael Haslam, Jason Erb, Philip Brown, Wayne Levin, Predictum Inc.

Introduction

A JMP add-in is a customization of JMP that is conveniently packaged as a single file with a .jmpaddin extension. Add-ins generally package one or more scripts along with structural information needed by JMP for the install process. As opposed to simple JSL files, which users can launch from the Finder or Windows Explorer, think of add-ins as a package of publisher-enabled and publisher-controlled functionality.

Publishers of add-ins include:

- Single users targeting their own purposes with their own functionality.
- Internal experts or groups of experts who want to distribute standardized functionality.
- Third parties offering licensed or custom software applications.

Benefit	Impact
Consistency	Everyone will have the same functionality.
Convenience	Scripts are easily deployed.
Security	Add-ins can be protected so that they cannot be edited; some distribution methods are secure as well.
Integration	Add-ins can look and feel as if they were part of standard JMP.

Table 1. Add-in benefits.

This paper is aimed at IT managers tasked with supporting and deploying JMP add-ins. IT may not be responsible for creating the add-ins, but it may be called upon to facilitate the distribution of add-ins and troubleshoot situations where things do not go as expected. This paper will delve into how add-ins are created and discuss distribution options and their risks. Sidebars throughout this paper offer tips, cautions and considerations relevant to each section.

To understand how best to distribute JMP add-ins, IT managers need to understand how they are structured, initially installed and loaded when JMP launches.

Finding JMP Add-Ins

Outside of your organization, a wide variety of innovative and useful add-ins can be found throughout various JMP communities, including Predictum's website (predictum.com/addins) and the JMP file exchange (jmp.com/community). Giving JMP users the ability to install and access add-ins can provide benefits in innovation, productivity and creativity.

Figure 1 summarizes what is covered in this paper. The three areas are interrelated; your knowledge of one will be incomplete without understanding the other two. This paper will cover add-in structure first, followed by distribution, installation and loading of JMP add-ins.

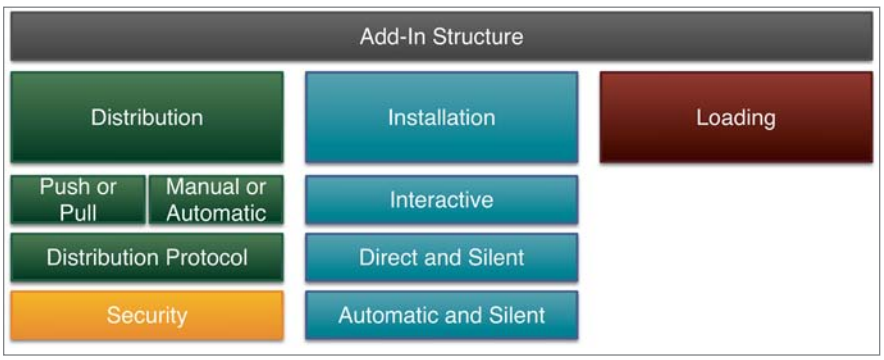

Figure 1. Main areas of add-in management.

Topic	Definition
Distribution	Distribution of add-ins, whether broadcast from a central source, pulled from a central source, or point-to-point.
Installation	Initial installation of add-ins to an individual user's JMP installation.
Loading	Loading add-ins that have been distributed and installed during the JMP launch process.

Table 2. Terminology.

The JMP® Add-In Structure

A JMP add-in is typically a package or ZIP file with the extension changed from .zip to .jmpaddin. This extension is registered to JMP during the initial JMP install. Table 3 shows the contents of a JMP add-in package.

Component	Comment	Role
addin.def	Fixed name. Mandatory file.	Registers the add-in within JMP. The add-in must have a unique name. Typically the name is in reverse DNS form: com.companyname.addinname.
addin.jmpcust	Fixed name. Optional file, not required if functionality is not menu-driven.	Defines where the add-in(s) are to be located on the JMP menu.
Content	Most often JMP Scripting Language (JSL) files – but can be any type of file, whether or not launched by JMP. An optional addinload.jsl file can be run at startup.	Provide functionality and/or content. For example, pictures can be incorporated into the add-in that are added to JMP reports it produces.

Table 3. Add-in components.

addin.def files are mandatory.
Errors in the content of this file and others in the add-in may cause installation problems. To examine these files, change the extension from .jmpaddin to .zip and examine the contents with any file compression utility.

All add-ins must contain the definition file (addin.def) that contains the name of the add-in. Other files are optional. If JMP scripts are found in the add-in package, they are stored locally in a named file structure in the JMP applications directory on Windows and the Application Support directory in the Users Library on Macintosh.

Add-ins, whether they appear as an item in a customized JMP menu or operate behind the scenes, can refer to any file in any location, local or network; they are not restricted to the add-in directory. Add-ins can be one or more of the following:

- A menu customization that runs a script installed in the JMP applications directory (Windows) or Application Support directory (Macintosh).
- A menu customization that runs a script on any local drive or network drive.
- A script in the JMP applications directory (Windows) or Application Support directory (Macintosh), local drive or network location that is run on JMP startup without involving a menu.
- A script in any location that is referenced in other JMP scripts or add-ins without involving a menu.
- Any sort of file located anywhere that may or may not be launched in JMP – again, without involving a menu.

This ability to point to and run scripts from any location has implications for the overall management of add-ins, including installation, distribution and control.

Add-In Distribution, Installation and Loading

IT managers need only concern themselves with distributing, installing and loading add-ins. They are generally not involved in creating add-ins.

Distribution Options

JMP add-ins can be distributed in either a push or pull manner and in either an automated or manual way. These distribution options, together with the installation and control methods, define the deployment scheme for add-ins.

	Manual Installation (User Initiates Install)	Automatic Installation
Pull	User pulls a package (ZIP) from a network or SharePoint location, FTP server or HTTP site. User then installs the add-in.	Add-in manager located on user PCs pulls a package (ZIP) or named add-in directory from a network or SharePoint location, FTP server or HTTP site. Add-in manager installs (ZIP) or JMP installed (named directory).
Push	Push a package (ZIP) to the user's hard drive or through email. Notify the user to double-click on the package to install.	1) Push a named add-in directory to the applications folder (All Users, Local). JMP installs automatically. 2) Push a named add-in folder or package (ZIP). Local PC add-in manager identifies and installs.

Table 4. Distribution options.

The push can be by any automated method that is normally used to deploy files across the organization. An add-in manager in the form of a JMP script (either home-grown or purchased from a third party) is typically written to handle the check and install tasks. An add-in manager can be written to check one or more add-in repository locations, pull down files, and either silently install or ask the user to authorize installation. JMP does not come with an add-in manager; one would have to be constructed or procured from a third party.

Installing Add-Ins

After an add-in is made available, there are three ways to install them.

	Installation Method	Comment
1	Interactive	The user opens the add-in from JMP (File > Open) or double-clicks on it. JMP will ask to (re)install the add-in (install if it's the first time the add-in is being installed, re-install if there's an add-in currently installed by the same name in the addin.def file).
2	Directed and silent	JMP is told to install the add-in through a JMP script. Installation occurs when this install script is run.
3	Automatic and silent	At startup, JMP sees and installs the new add-in in the JMP local or all users application directory (see below). Installation occurs when the user next launches JMP.

Table 5. Add-in distribution and installation.

The interactive and directed install processes unzip a single add-in package (ZIP file) and place the files in a named folder in the JMP local applications directory (Windows) or Application Support directory (Macintosh). The folder name in the local applications directory is the name given to the add-in the in addin.def file. This is typically a unique name in reverse DNS format (com.companyname.addinname). Thus, the addin.def with a unique name is the only information required by JMP for an add-in to be installed. Any menu customizations found in the package are also applied. If a file with the name addinload.jsl is found, then this file will be run immediately.

The automatic install process looks in the JMP "all users" or "local users" application directories to see if they contain new addin.def files. The "all users" folder is a shared folder for all accounts on the same PC or Mac. The "local users" folder is nested under the directory structure for each user with an account on the PC or Mac. (Refer to Distribution Options on page 4.)

These files are processed to install the add-ins specified in those definition files. This procedure is simply a part of the process JMP uses to install add-ins when it first starts up (see Add-In Loading During JMP Startup on page 6). This option is automatic and silent. You can create a new add-in, place it in one of the applications folders as an addin.def (and, optionally, other files included), and JMP will install it. Menu customizations, if present, will be applied, and the addinload.jsl file will be run if included in the add-in folder.

Problems with the installation? Here are some areas to look at:

- Check the install directories to see if the add-ins were successfully distributed.
- Check the addin.def file to ensure that it is not overwriting an add-in with the same name. The addin.def file should be opened with caution. Ensure that no hidden/special characters contaminate the file.
- Though the .jmpcust file is in XML format, it is best to only use JMP to create or edit it. A direct edit often leads to problems.

Add-In Loading During JMP® Startup

At startup, JMP will reinstall all previously installed add-ins as registered in the addinRegistry.xml file found in the local applications directory (Windows) or ~/Library/JMP (Mac). Each time JMP is launched, add-ins in the JMP applications directories for the local user and all users are identified where:

- Previously installed add-ins are reinstalled and their menu customizations are reapplied. The addinload.jsl file is run, if present.
- Newly identified add-ins trigger JMP to install the add-in.

The startup install process is sequential. Already known and previously installed add-ins from the XML registered add-ins file are installed first in alphabetical order by add-in name. New add-ins are then identified and installed, first by folder and then file within the folder. Add-ins found in the all-users location are installed first, followed by the local-users location. The alphabetical order of the add-ins in the specific location being used is the install sequence from that location. Note that in JMP 11 load order for registered add-ins can be specified in the View > Add-Ins menu.

JMP logs issues associated with loading add-ins in its log window.

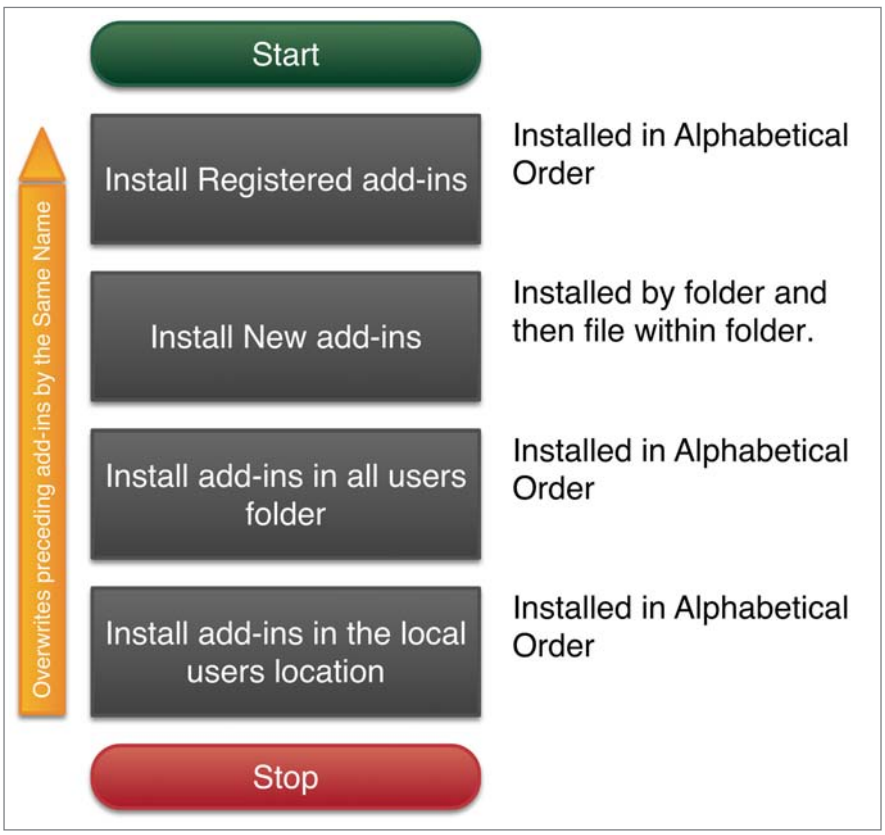

Figure 2. Sequential add-in loading during JMP startup.

Add-in file handling

The last add-in installed will overwrite files and menu customizations from earlier installs. Additionally, an add-in with the same name installed later might overwrite some files but perhaps not all earlier files. If you have an add-in with the same name in both all-users and local-users directories, the local users version will take precedence.

Access By	Location
All users on the machine	%ALLUSERSPROFILE%\SAS\JMP\AddIns
By account name	%LOCALAPPDATA%\SAS\JMP\AddIns
By account name	%USERPROFILE%\Local Settings\Application Data\SAS\JMP\AddIns (Windows XP)

Table 6. File location on Windows.

Access By	Location
All users	/Library/Application Support/JMP/Addins
By account name	~/Library/Application Support/JMP/Addins

Table 7. File location on Macintosh.

The installation process for add-ins performs no locking operations on the files associated with the add-in, and files are not copied or sequestered when JMP loads. The add-in files can be replaced at any time. The addinload.jsl file will allow code to be run immediately on the loading of a specific add-in as JMP starts up. This has implications for control of add-ins.

The Add-In Home Folder

Each add-in specifies its own home folder under the Addins folder. The name of the folder is the name identified in the addin.def file. So, for example, an addin.def file that has the line:

ID = com.predictum.modellingaids

will find the content of the add-in in a folder under /Addins/com.predictum.modellingaids.

The assumption is that the executable JSL and other content are within the same home folder – but they do not have to be. Including a Home = statement in the addin.def file can set the home folder to anything local or over the network.

Cautions, Control and Compliance

Since the add-ins feature was added to JMP, a wide variety have become available on JMP's website, as well as third-party websites such as predictum.com, and they have been exchanged via social media. Standard JMP installations allow any JMP user the ability to install add-ins at will. This can lead to problems, particularly with respect to ensuring control and compliance.

Add-ins and upgrading JMP®

The registry and the add-ins folder are common across all installed versions of JMP, so upgrading JMP is independent of upgrading the add-ins. This can lead to some problems when add-ins are launched from menus.

Considerations for control of add-ins are:

- Do you need to have control of all add-ins or just some add-ins?
- Do you need version control? Does the version need to be verified each time the add-in is run to be sure that the user is running the correct version?
- Can a user run a particular add-in while not connected to the network?

The script component of JMP add-ins can be encrypted, ensuring that its contents are not tampered with. The add-in structure is a different story. The issues to consider are:

- Menu customizations can be changed by the end user, effectively pointing to an alternate script from the one intended.
- Scripts in the applications directory can be replaced.
- The registry of add-ins can be changed.
- An add-in can be installed that overwrites an existing add-in.

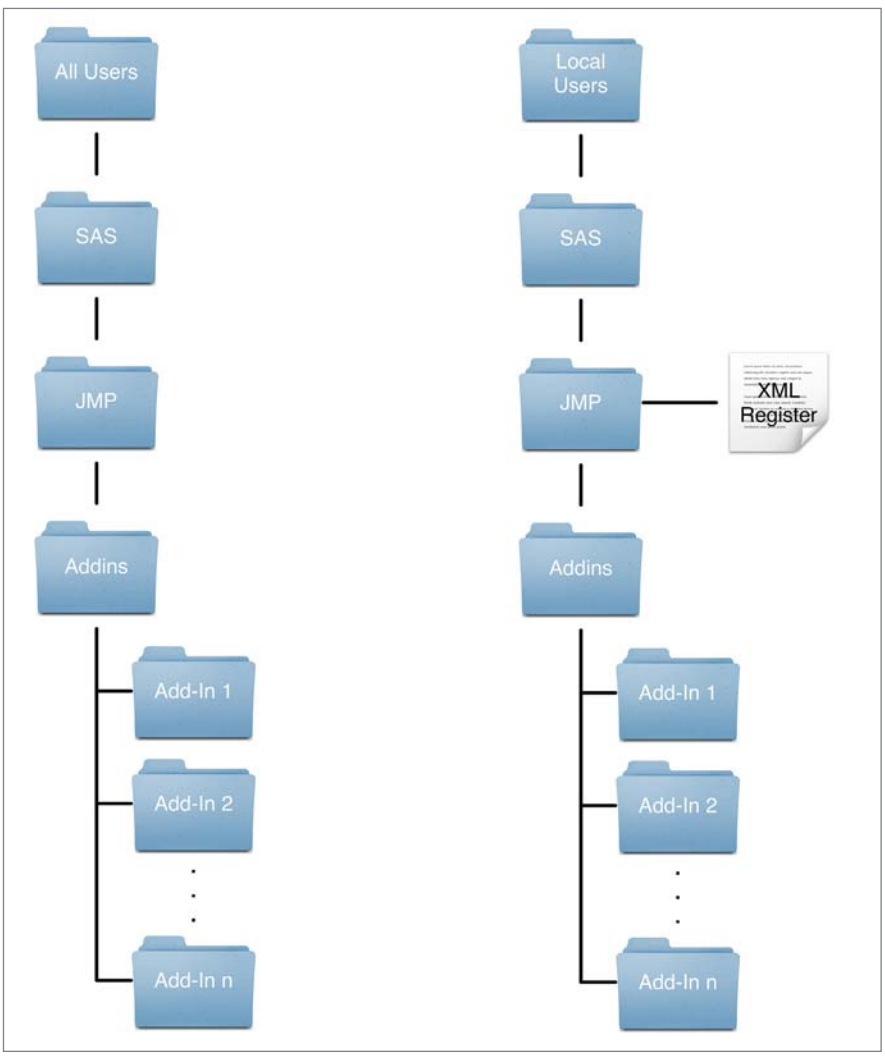

Figure 3. Relative location of add-in files.

Figure 3 shows the relative location of the add-in files, labeled Add-In 1, Add-In 2, etc. The register of add-ins maintained by JMP is shown in the local user applications directory. Based on this structure, there are five methods for maintaining control of scripts and ensuring compliance:

- Lock down via access control for one or both of the JMP applications directories. Locking down the local user JMP applications directory has the consequence of making it impossible to install additional add-ins without privileges. JMP must have the privileges to modify its local applications directory for add-ins to be installed from packages; create a directory; copy files; and modify the XML register. The all-user JMP application directory can be locked down without having this effect.
- Lock down an individual add-in directory within the JMP applications directory. This allows an individual add-in to be controlled while allowing the user to install additional add-ins.
- Create encrypted scripts that check and verify their version when run. Checksum methods and querying a network source for version information can be used effectively to accomplish this.
- Run add-in scripts from a secure network location.
- Create a master script that checks the versions of each controlled add-in and updates as needed. This script can be restricted, if needed. This function can also be facilitated by an add-in manager, as discussed previously.

Security and Automation

Deployment considerations include how best to distribute the add-ins, network type, security issues, degree of automation and end-user control of add-ins. The following two tables summarize the decisions to be made.

		Will end user be allowed to install add-ins?	
		Yes	**No**
Are any add-ins required to be strictly controlled?	No	Lock all user add-ins or lock specific folders on local users' PCs (risk: low).	Lock local add-ins and XML add-in registry (addinRegistry.xml). Lock all-users folder (risk: moderate).
	Yes		Lock local add-ins and XML add-in registry (addinRegistry.xml). Lock all-users folder (risk: moderate).

Table 8. Add-in control.

Some organizations need to tightly control at least some of their add-ins. Doing so requires, as indicated in the upper-right quadrant of the above table, that you lock the user's local folder and all-user folders since, as discussed above, add-ins overwrite others of the same name. The XML add-in registry will also need to be locked. On the opposite end of the spectrum, if users are allowed to install their own add-ins and there is no need to strictly control add-ins, then, as indicated in the lower-left of the above table, no action is required.

		Automatic Installs & Updates	
		Yes	**No**
Active network connection to run add-ins?	No	Employ an add-in manager. Store add-in files locally (risk: low).	User manually installs add-ins locally (risk: moderate).
	Yes	Choose: 1) Create network location for add-ins and push updates. 2) Employ an add-in manager. (risk: moderate).	Lock local add-ins and XML add-in registry (addinRegistry.xml). Lock all-users folder (risk: moderate).

Table 9. Automation and network connectivity.

Add-ins are already very convenient ways of distributing functionality. You may want to add to this convenience by making add-in distribution automatic. The ultimate way to do this is to develop or acquire an add-in manager as described under "Yes" for "Automatic Installs & Updates" in the above table. An alternative is to do this by way of shared network folders, but a word of caution: Our experience with this approach is uneven. Network latency issues often scuttle the transfer and install of the add-ins. As a result, you cannot be sure that the transfer occurred successfully. An add-in manager can monitor the transfer; retry if it experiences trouble; warn if transfers were not completed; and log the who, what, when and where of a distributed network of add-ins.

Those interested in compliance, convenience or both should consider an add-In manager. But those whose needs are less demanding can simply make add-ins available over the network and/or have add-ins installed locally that point to JSL files and other resources over the network.

Add-In Repositories: Where to Locate Add-Ins for Distribution

An add-in package is a ZIP file with a .jmpaddin extension instead of a .zip extension. Any standard file compression utility, whether built into the OS or a third-party utility, can create, examine and expand JMPADDIN files as they do ZIP files. And just like ZIP files, JMPADDIN files can be stored on a local hard drive, network or SharePoint location, FTP server or HTTP server.

Deployment Method	Login	Security	Comment
FTP Server	User account or anonymous (no account setup).	Can enable tracking of who downloaded what by enabling web server logging.	Widely available, standard configuration on Windows and Mac.
Network Location	Requires user account.	Governed by OS security policies.	Tied to a specific OS, not as easy to use.
HTTP Site	User account or anonymous (no account setup).	Can enable tracking of who downloaded what by enabling web server logging.	Special setup required for automated downloads.

Table 10. Distribution considerations.

Deployment Methods

FTP server allows for anonymous or user login. The advantage of anonymous login is easy distribution and download of the add-ins; there's no need to set up accounts for each JMP user, but there's no tracking of who downloaded what. Tracking downloads requires setting up accounts – but that also improves security.

Otherwise, downloading add-ins via FTP is attractive because it's simple, widely supported and globally accessible. FTP servers are included by default on the Windows Server editions and on the Mac OS X server, and are relatively easy to configure. Windows and Mac operating systems include FTP clients by default, so there's no need to provide any additional end-user software.

Network download requires a user account on the operating system-specific network. The main advantage of network downloads of add-ins is security. Access to the add-ins will be governed by the operating systems' security policies. Users can be assigned privileges based on grouping characteristics that are generally easy to configure. The disadvantages of network downloads of add-ins include being tied to a specific operating system and difficulty of use.

HTTP download allows for anonymous or user login. The main advantage of HTTP downloads of add-ins is ease of use due to widespread support and global download capabilities. A free HTTP server can be set up on a Windows or Mac server, and all Windows and Mac clients come with a default web browser. Most users are familiar with a web browser, and that makes this option the most simple and versatile.

The security of the HTTP download approach depends on whether a secure connection is established via HTTP. Without an encrypted connection, anyone can intercept the proprietary add-in code. A secure HTTP connection is recommended. Special setup or software may be required to facilitate automatic downloads.

Cautions and Considerations

Customizing the Menu

When JMP starts up, add-ins are installed in a set order, and their menu customizations are applied in that same order. This order is discussed in the installation section of this document. Because personal computer style menus are hierarchical in nature, the order in which menu items are added can affect how the menus display.

To install a new add-in where there are other add-ins already installed while maintaining a particular order, you have to uninstall all add-ins and then reinstall them in the desired order.

Dependencies can get you in trouble, so best practice suggests that the entire menu structure, including higher levels of the menu structures linking up to standard JMP, be specified. If a new add-in is anchored to a menu item that is not present, JMP will likely locate it at the bottom of the next-highest, identifiable menu.

Keep two things in mind. First, the menu structures for Macintosh are not identical to Windows. Second, new versions of JMP may involve changes in the menu structure. The best way to handle these differences is to use JMP software's built-in functions for editing menus (File > New > New Add-In). Do not edit the JMP .jmpcust XML files. It's very easy to mess it up. For example, anchoring an add-in below Help > About JMP will work fine on Windows because that menu item is present. But About JMP is found under the JMP menu on Macintosh. In this case JMP will place the new add-in at the bottom of the Help menu.

No Menu Add-Ins

JMP add-ins can be created that do not involve customizing the JMP menu. Perhaps the most common menu-less add-in is one that provides a function or functionality not found natively in JMP.

Unlike typical JMP add-ins, menu-less add-ins provide functionality that is not typically attached to a particular function or procedure. Indeed, they may provide functionality that other add-ins take advantage of. While they can be loaded like regular JMP add-ins, you might want to consider adding the functionality by way of a JMPstart.jsl file.

JMPstart.jsl files, as the name suggests, are run at the time JMP is launched. Functions, expressions, global variables, etc., created in the JMPstart.jsl file are evaluated and loaded into memory as JMP starts up.

While JMPstart.jsl is meant for individual users, a similar file, JMPstartAdmin.jsl, is meant for all JMP users on a particular machine. The section “Running a Script at Startup” in Chapter 9 of the JMP Scripting Guide details where to locate both JMPstart.jsl and JMPstartAdmin.jsl files.

JMPstart

```
1  //Open a user-tracking file and a
2  dt = Open( "http://www.predictum.com/download/products/addin_whitepaper/user_log.jmp" );
3  dt << add rows( 1 );
4  Column( dt, "Date" )[N Rows( dt )] = Today();
5  If( Host is( mac ),
6      Column( dt, "User Name" )[N Rows( dt )] = Get Environment Variable( "USER" ),
7      Column( dt, "User Name" )[N Rows( dt )] = Get Environment Variable( "username" )
8  );
9  //close (dt);   //for illustration, the file will be left open and will not update
10
11
12 //Calculate mode function
13 fn_mode = Function( {matrix_input},
14     {local_matrix_input = matrix_input},
15     max_count_index = .;
16     max_count = 0;
17     For( i_row = 1, i_row <= N Rows( local_matrix_input ), i_row++,
18         current_count = N Rows(
19             Loc( As List( local_matrix_input - local_matrix_input[i_row] ), 0 )
20         );
21         If( current_count > max_count,
22             max_count = current_count;
23             max_count_index = i_row;
24         );
25     );
26     //mode_count = max_count;
27     mode_output = num_matrix[max_count_index];
28     m_output = V Concat( Matrix( mode_output ), Matrix( max_count ) );
29 );
```

Figure 4. Adding a new function to calculate the mode of a set of numbers.

As an example, let’s add functionality to track who is using JMP by adding a new function to calculate the mode of a set of numbers (Figure 4).

Locate the file in the users local directory. A JMP data table, serving as a log, will open and add the current user’s name each time JMP is launched (the file will not save in this illustration). Additionally, a new function for calculating the mode of a matrix is available. To test it, open a script window or the log and enter:

```
num_matrix = [5, 2, 3, 2, 2, 2, 7, 5, 7, 7, 3];

fn_mode(num_matrix);  //returns a matrix [mode value, mode count]
```

Platform Considerations and Installation Errors

If the add-ins are Windows- or Mac-specific, this should be specified through the “host” parameter of addin.def. If the add-in is Windows-specific, the “host” parameter should be “Win” (host = “Win”), and if it’s Mac-specific, it should be “Mac”. Add-ins should be certified to work on specific platforms, and not just assumed.

New menu item placement will depend on the built-in Windows- and Mac-specific menu items. It is therefore recommended that add-ins only add new menu items in relation to shared menu items on both Windows and the Mac.

Keep in mind that menus tend to change as new versions of JMP are released.

When add-in anchors are missing, the add-ins will be added as close within the hierarchy as possible. For example, an add-in anchored to About JMP in the Help menu on Windows will appear at the bottom of the Help menu on Mac because there is no About JMP item on the Mac Help menu.

JMP posts errors in the log whenever it encounters problems installing add-ins (Figure 5). Menu placement problems, name collisions, XML errors and the like associated with add-ins will be reported.

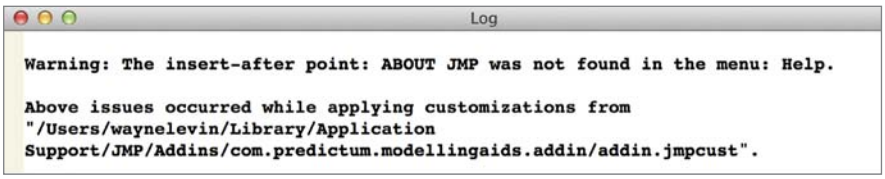

Figure 5. Sample installation problem documented in error log.

For those doing cross-platform add-in distribution, a best practice is to direct add-ins to shared menu items on the Windows and Mac platforms – File, Tables, Graph, View, Window, Help – so they work in both environments.

Uninstalling Add-Ins

Removing add-ins can be a bit deceptive.

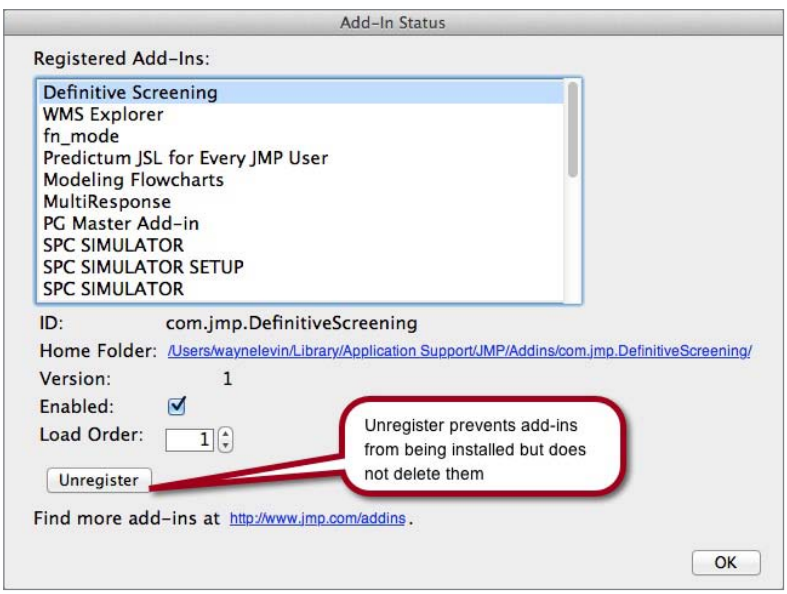

Figure 6. Removing add-ins.

To remove an add-in, go to View > Add-Ins to see a list of currently installed add-ins. To remove an add-in, select it and click the “Unregister” button. However, the add-in content remains in its folders (see the Installation section). To fully remove an add-in, you must, therefore, remove it from the content manually or with an add-in manager.

Cautions for Editing addin.def Files

Some text editors may be unintentionally helpful by adding hidden characters that will disrupt installing the add-in.

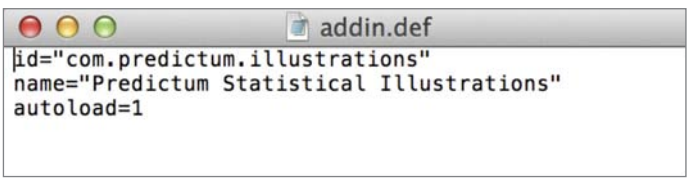

Figure 7. Sample of simple addin.def file.

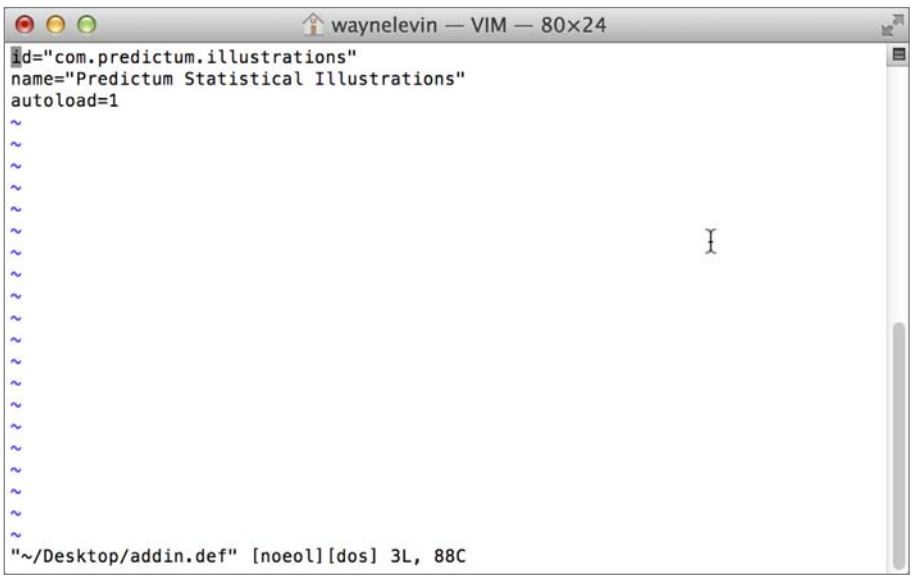

Figure 8. Sample of add-in using VIM on the Macintosh.

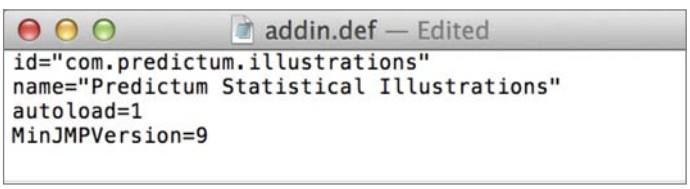

Figure 9. Adding a new line to the Mac addin.def file to specify the minimum JMP version.

Using Textedit on the Mac, one can add a new line to specify the minimum JMP version (Figure 9).

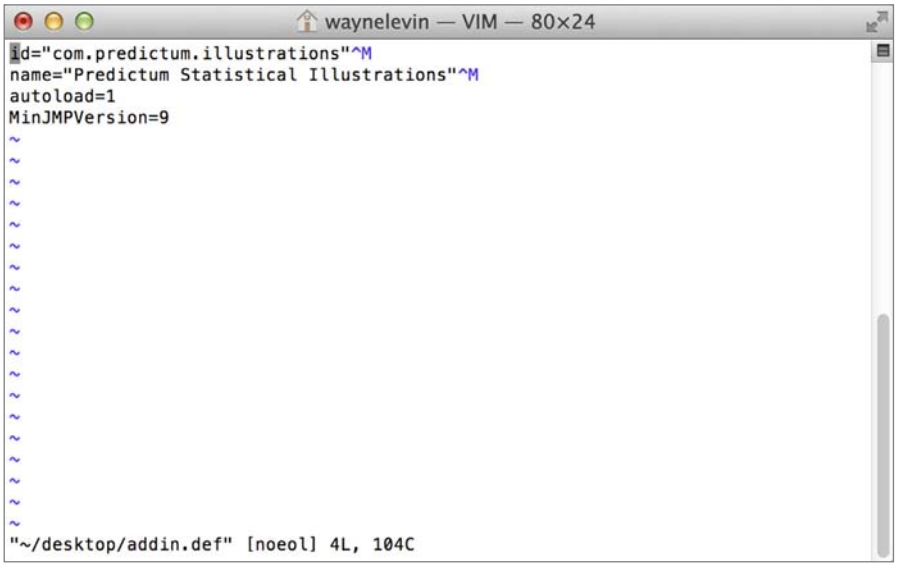

Figure 10. Sample of control characters added by cutting and pasting text.

Pasting text can sometimes introduce hidden characters to an addin.def file and prevent the add-in from loading. Note that in Figure 10 there are CTRL+M (^M) characters at the ends of the top two lines, but there is no evidence of them in Textedit (and other editors).

Appendix A. Creating Add-Ins

Creating add-ins is a multistep process that begins by manually creating an addin.def file (Figure 11) or using the JMP Add-In Builder (Figure 12) using File > New > Add-In. Instructions for creating add-ins can be found in the *JMP Scripting Guide* documentation: Help > Books > Scripting Guide.

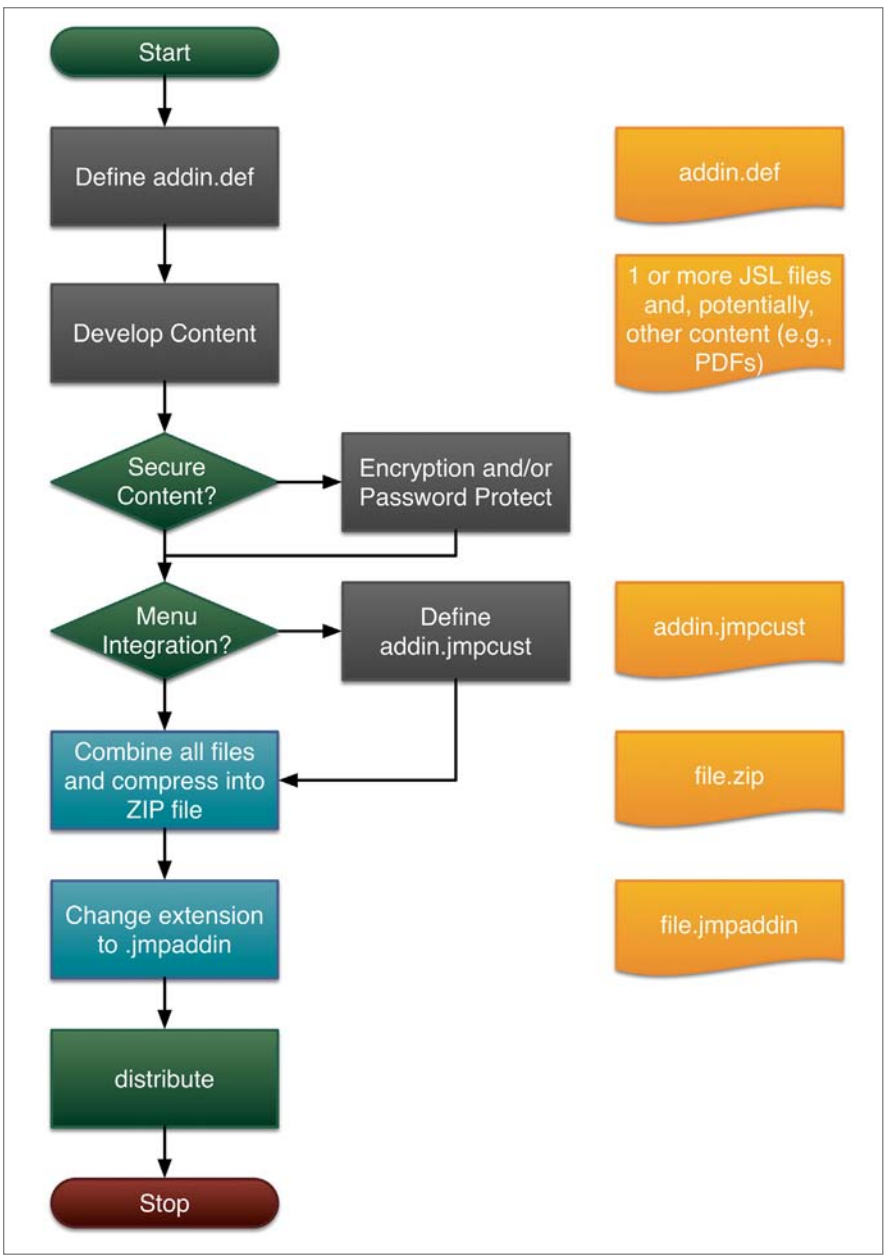

Figure 11. Steps for creating an add-in.

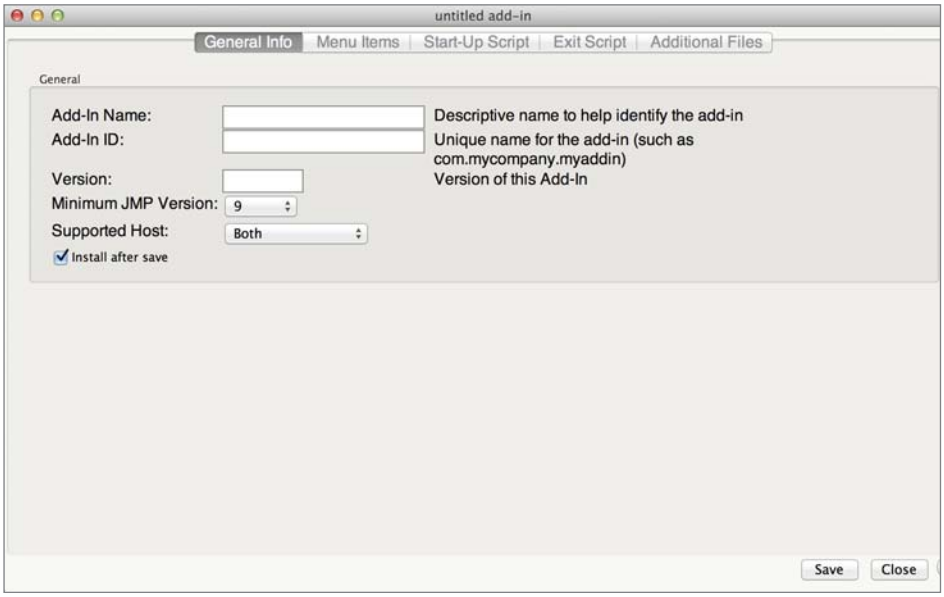

Figure 12. The JMP Add-In Builder.

Appendix B. addin.def Components

The addin.def file is a simple text file containing name-value pairs that provide registration information about a JMP add-in. Here are the name-value pairs to include in the addin.def file:

id

Required. The unique ID for your add-in. The string can contain up to 64 characters. The string must begin with a letter and contain only letters, numbers, periods, and underscores. Reverse-DNS names are recommended to increase the likelihood of uniqueness.

name

Optional. The name that can be displayed in the JMP user interface wherever add-in names are displayed, instead of the unique ID. This name is displayed if no localized names are provided or when JMP is run under a language for which you did not provide a localized name.

name_xx

Optional. Allows the user-friendly name to be localized for different languages, where xx is the two-letter ISO 639-1 code for the language. If you include localized names, you should still include a language-neutral name in case JMP is running under regional settings for which you do not have a localized name.

home

Optional. The path to the add-in files. The Home Folder for the add-in is assumed to be the folder where addin.def is located. You need to include a setting for home only if the Home Folder is somewhere else (for example, a network shared folder).

home_win

Optional. The path to the add-in files to be used when JMP is running on Windows. Overrides the value specified for home on Windows, if any.

home_mac

Optional. The path to the add-in files to be used when JMP is running on the Macintosh. Overrides the value specified for home on Macintosh, if any.

autoLoad

Optional, Boolean. The default value is True (1). Determines whether this add-in is initially configured to load automatically during JMP startup.

host

Optional. Valid values are Win and Mac.

minJMPVersion

Optional. Valid values are integers corresponding to the JMP major version that is the minimum version that the add-in supports. Note that this feature does not currently incorporate minor JMP revisions. If an add-in requires JMP 10.0.2, due to a bug fix or enhancement since JMP 10, this feature will trap those users running JMP 9.x.x but not those users running JMP 10.0.1 or JMP 10.

maxJMPVersion

Optional. Valid values are integers corresponding to the JMP major version that is the maximum version that the add-in supports. Use this setting only if there is a known incompatibility between your add-in and a specific version of JMP. You should provide a new version of the add-in for later versions of JMP.

About SAS and JMP

JMP is a software solution from SAS that was first launched in 1989. John Sall, SAS co-founder and Executive Vice President, is the chief architect of JMP. SAS is the leader in business analytics software and services, and the largest independent vendor in the business intelligence market. Through innovative solutions, SAS helps customers at more than 70,000 sites improve performance and deliver value by making better decisions faster. Since 1976 SAS has been giving customers around the world THE POWER TO KNOW®.

SAS Institute Inc. World Headquarters +1 919 677 8000

JMP is a software solution from SAS. To learn more about SAS, visit **sas.com**
For JMP sales in the US and Canada, call 877 594 6567 or go to **jmp.com**