

InstallJammer

InstallJammer 1.2 User Guide



by Damon Courtney <damon@installjammer.com>

Table of Contents

InstallJammer 1.2 User Guide	8
Welcome	8
InstallJammer Help	9
Getting Started	9
Frequently Asked Questions	10
The Install Builder	12
Getting to Know the Install Builder	12
Install Builder Preferences	13
Install Builder Command Line Options	15
General Information	17
Application Information	17
Platform Information	22
Package and Archive Information	26
Components and Files	28
Groups and Files	28
Files and Directories	31
Components	34
Setup Types	36
User Interface and Events	38
Panes and Actions	38
Command Line Options	40
Virtual Definitions	43
Virtual Text	43
Run Disk Builder	44
Disk Builder	44
Test the Installation	45
Test Installer	45
Test Uninstaller	46
Objects	47
Object Types	47
Standard Properties	48
Panes	49
What are Panes?	49
Actions	51
What are Actions?	51
What are Action Groups?	52
Standard Action Properties	53
Console Actions	54
Console Ask Yes or No	54
Console Clear Screen	55
Console Get User Input	56
Console Message	57
Console Pause	58
Execute Actions	59
Execute Action	59
Execute External Program	60
Execute Script	63
File Actions	64
Adjust Line Feeds	64
Backup File	65

Change File Ownership	67
Change File Permissions	68
Copy File	69
Create File Link	70
Create Folder	71
Delete File	72
Read File Into Virtual Text	73
Rename File	74
Replace Text in File	75
Write Text to File	76
Unzip File	78
General Actions	79
Continue Install	79
Exit	80
Fetch URL	81
Generate UID	82
Launch File	83
Launch Web Browser	84
Log Debug Message	85
Message Box	86
Message Panel	87
Modify Object	89
Pause Install	90
Set Object Property	91
Set Virtual Text	92
Stop Install	93
Text Window	94
Wait	95
Install Actions	96
Add Install Info	96
Add Response File Info	97
Add to Uninstall	98
Check for Previous Install	99
Get Previous Install Info	101
Install Log File	103
Install Selected Files	104
Install Uninstaller	105
Install Wish Binary	106
Install Wrapped Script	107
Unpack Stored File	109
Java Actions	110
Get Java Property	110
Locate Java Runtime	111
Shortcut Actions	113
Install Desktop Shortcut	113
Install Program Folder Shortcut	115
Install UNIX Program Folder	117
Install UNIX Shortcut	118
Install Windows Shortcut	120
System Actions	121
Add Directory to Path	121
Add Environment Variable	122
Delete Environment Variable	123
Reboot or Shutdown System	124

Register Package	125
Remove Directory from Path	127
Uninstall Actions	128
Uninstall Leftover Files	128
Uninstall Selected Files	129
Windows Actions	130
Disable Wow64 Redirection	130
Revert Wow64 Redirection	131
Windows Registry Actions	132
Add Windows File Command	132
Add Windows File Extension	133
Add Windows File Type	134
Add Windows Registry Key	135
Add Windows Uninstall Entry	136
Import Windows Registry File	138
Register Windows Library	139
Remove Windows Registry Key	140
Unregister Windows Library	141
Windows Service Actions	142
Continue Windows Service	142
Create Windows Service	143
Delete Windows Service	146
Pause Windows Service	147
Start Windows Service	148
Stop Windows Service	149
Wizard Actions	150
Add Pane to Order	150
Add Widget	151
Append Text to Widget	154
Create Install Panes	155
Destroy Widget	156
Focus On Widget	157
Modify Widget	158
Move Forward	159
Move to Pane	160
Populate Components	161
Populate Setup Types	162
Remove Pane from Order	163
Conditions	164
What are Conditions?	164
Standard Condition Properties	165
File Conditions	166
File Exists Condition	166
File Name Test Condition	167
File Permission Condition	168
General Conditions	169
Command Line Test Condition	169
Execute Script Condition	170
Object Test Condition	171
Script Condition	172
Virtual Text Test Condition	173
Platform Conditions	174
Platform Condition	174
String Conditions	175

String Equal Condition	175
String Is Condition	176
String Match Condition	177
System Conditions	178
Env Variable Test Condition	178
Package Test Condition	179
Port Test Condition	180
User Input Conditions	181
Ask Yes or No	181
Windows Conditions	182
File Extension Test Condition	182
File Type Test Condition	183
Registry Test Condition	184
Windows Service Test Condition	185
Virtual Text	186
What is Virtual Text?	186
Virtual Text Definitions	188
Virtual Directory Definitions	196
Developer's Guide	201
Debugging an Install	201
Builder API	202
What is the Builder API?	202
Builder API Calls	203
GetAction	203
GetActionGroup	204
ModifyObject	205
SetPlatformProperty	206
Install API	207
What is the Install API?	207
Install API Calls	209
AddInstallInfo	209
AddLanguage	210
CommandLineAPI	211
ComponentAPI	212
ConfigAPI	213
CopyObject	214
ErrorMessage	215
DestroyWidget	216
Exit	217
Fetch URL	218
FindProcesses	219
GetComponentsForSetupType	220
GetInstallSize	221
GetSelectedFiles	222
GetSystemPackageManager	223
GetWidgetChildren	224
GetWidgetPath	225
EnvironmentVariableExists	226
FindObjects	227
LanguageAPI	229
LoadMessageCatalog	230
PromptForDirectory	232
PromptForFile	233
PropertyFileAPI	235

ReadInstallInfo	237
ResponseFileAPI	238
RollbackInstall	239
SetActiveSetupType	240
SetExitCode	241
SetFileTypeEOL	242
SetObjectProperty	243
SetVirtualText	244
SubstVirtualText	245
URLIsValid	246
VirtualTextAPI	247
VirtualTextExists	248
Tutorials	249
Create a New Install Fast	249
Create a New Install Step-by-Step	250
Tcl/Tk Help	258
What is Tcl/Tk?	258

InstallJammer 1.2 User Guide

Welcome

InstallJammer is a multiplatform GUI installer and builder designed to be completely cross-platform and function on Windows and most all versions of UNIX with eventual support for Mac OS X.

Using InstallJammer

You can read through [Getting Started](#) to learn a little bit about how to start using InstallJammer.

New Features and Bug Fixes

If you are upgrading from a previous version of InstallJammer, read through the [Release Notes](#) to learn about all of the changes since the last version.

Helping InstallJammer Development

InstallJammer will continue to develop with help from people like you. Please report any bugs that you find, or let us know if there are features that you would like to see in InstallJammer. InstallJammer is always improving and still has bugs to work out, but it can only get better with help from you. Below are some links that may help you help us.

[InstallJammer Homepage](#)

[Ask a Question](#)

[Report a Bug or Request a New Feature](#)

-0-

InstallJammer Help

Getting Started

The easiest way to get started using InstallJammer for the first time is to read through some of our tutorials. If you have a good grasp of installers and how they work, you can probably give a quick read through [Create a New Install Fast](#) to get started.

If you want a more detailed look at building a new install, read through the [Create a New Install Step-by-Step](#) tutorial.

If you're already familiar with how to use InstallJammer, or you just want to learn something maybe you didn't know already, you can read through our [Frequently Asked Questions](#) to see if maybe what you're looking for is in there.

Once you have your first install project built, read through [Getting to Know the Install Builder](#). It will give you a good introduction to InstallJammer's install builder and how to use it effectively.

After you finish learning your way around InstallJammer's install builder, read a little bit more about [Virtual Text](#) and its use within InstallJammer. You might also want to read through [What is Virtual Text?](#) and the list of [Virtual Text Definitions](#) or [Virtual Directory Definitions](#).

Once you understand virtual text a little better, it's time to learn about what the customer really sees. The panes of your install. Read up on [What are Panes?](#) to become more familiar with how InstallJammer displays the installer to the user.

When you think you understand the basics of installs and the use of virtual text, you might want to read through [What are Actions?](#) Actions are used throughout the install process, and it's worthwhile to read up on what they are, how you use them and what types of actions are available.

After reading about actions, you'll probably want to know how to control those actions. Read up on [What are Conditions?](#) to get a feel for how InstallJammer controls the flow of your installer.

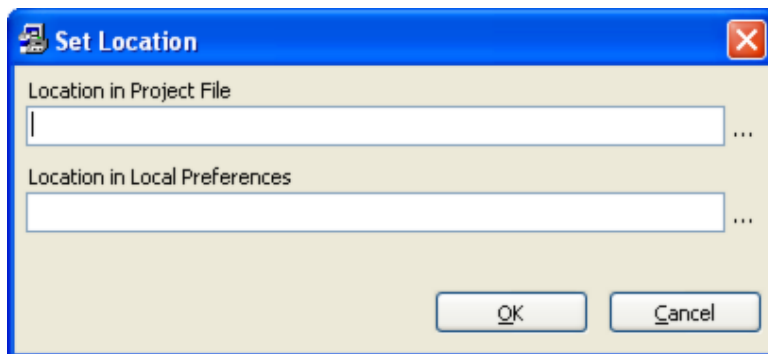
-0-

Frequently Asked Questions

1. How do I save relative paths for the files in my install project?

Most installers give you the option of a check box that allows you to save all of the files in your project as relative paths. Relative usually being relative to the location of the project file itself. This is an extremely useful feature since most developers don't usually put their work in the same locations on their own machines.

Instead of just a simple check box for saving paths as relative, InstallJammer allows you the ability to completely re-map the location of any file in your project. This is done through the Location property for directories and files. If you select a file or directory in your project, you will see a standard property called Location. This property tells InstallJammer where to find this file or directory on the local system. If you click the Location property, you see a browse button that opens a popup for specifying the location.



InstallJammer lets you specify the location as a directory that can be stored back into the project file itself or one that you can store in your own local copy of your InstallJammer preferences.

If the location is specified for a directory, it is the actual location of that directory, not the directory that contains that directory. If the location is specified for a file, it is the directory that contains that file.

Locations can be specified as a full directory path, a relative directory path or virtual text. A relative directory path is relative to the location of the project file itself. Virtual text can contain a full or relative path and will be substituted at build time to get the file location.

For example, you could specify the Location of your main application directory as `<%BaseDir%>`. Then, in your Virtual Text Strings table, create a new virtual text variable BaseDir with the location of your application. Then, you can actually build from the command-line with a different location by specifying `-DBaseDir /path/to/your/application`.

2. Where is the best place to put my project file so that it is part of my application directory?

Many developers wish to put their InstallJammer project files somewhere within their application structure so that the project file can be easily included when packaging the application itself and so that the project file can be maintained in the same source repository as the application itself.

It's best to put your project file in its own subdirectory because InstallJammer uses the project file location as a base for creating a build/ and output/ subdirectory. So, the best place to put your project file so that it is part of your application directory is in a subdirectory

off the main application directory. An installer/ subdirectory would do nicely.

With this in place, you can easily specify the Location of your main application directory as ".." and have all of your application files be relative to your project file. Since relative Locations are relative to the project file, a Location of .. means that the files are located one directory down.

3. How do I build from the command-line with a different location for my application?

This is easily done by giving the directories in your install project a virtual text Location. If the Location of a directory is specified as virtual text, you can then specify a new value for that virtual text on the command-line when you build.

Let's say you have a main application directory, and you specify the Location of that directory as <%BaseDir%>. Then, in your project file, you can create a new virtual text variable called BaseDir with a value of /my/application/is/here. Now, you want to build from the command-line with a different directory location for your application, so you do:

```
installjammer -DBaseDir /path/to/my/application /path/to/my/project.mpi
```

The -D option lets you specify a virtual text variable and a value that will be overridden in the install project before the installer is built. The last argument on the command-line is always the path to your project file. Using the above command, your project will be built with your application files located in /path/to/my/application.

-0-

The Install Builder

Getting to Know the Install Builder

Read through the following sections to help you get to know the install builder a little better. Each section describes the different pieces of the install builder and how to use them.

-0-

Install Builder Preferences

Directories

Project Directory Location

This preference defines the location of your InstallJammer projects. You can choose to save your projects anywhere, but InstallJammer will use this property by default as a location for new projects when they are created.

Custom Theme Directory Location

This preference defines where InstallJammer will look for custom themes that you may have for your version of InstallJammer. The directory specified must contain a subdirectory for every theme, and the theme will be added to the list of available themes when creating a new project.

Custom Action Directory Location

This preference defines where InstallJammer will look for custom actions that you may have for your version of InstallJammer. The directory can contain any number of .action files that will then appear in the list of available actions under the category Custom Actions.

Custom Condition Directory Location

This preference defines where InstallJammer will look for custom actions that you may have for your version of InstallJammer. The directory can contain any number of .condition files that will then appear in the list of available conditions under the category Custom Conditions.

External Programs

File Explorer

This preference defines the default file explorer when using the explore features in InstallJammer. If one is not specified, InstallJammer will attempt to use the default file explorer for your system. On Windows, InstallJammer will always use Windows Explorer unless specified. On UNIX systems, InstallJammer will attempt to determine the best file explorer based on your window manager.

Help Browser

This preference defines the web browser you wish to use when browsing InstallJammer's help documentation. By default, InstallJammer will attempt to find the best browser based on your system environment and a list of acceptable browsers to InstallJammer. Almost any web browser should work properly.

Editor

This preference defines an external editor program to use when editing long text within the install builder. If an editor is specified, clicking the "..." button to edit a text field will save the text to a temporary file and then launch the external editor on that file. The builder will then watch the file for changes and repopulate the project with the new data. Note that InstallJammer will not create a console window for editing with a text-based editor.

InstallJammer Update

Check for updates to InstallJammer on startup

If this preference is true, InstallJammer will check for new versions of itself everytime it is started. If it is false, you must check for updates by yourself using the Check for Updates Now button on this tab.

Use a proxy server

If true, InstallJammer will use an HTTP proxy server when checking for its updates.

Proxy Host

The hostname for the proxy server to use.

Proxy Port

The port for the proxy server to use.

-0-

Install Builder Command Line Options

InstallJammer can be started with several command-line options that make building installs easier for some jobs. In most cases, these are used to provide quick builds of a single project or build projects using scripts and other programs.

Usage:

```
installjammer ?option option ... --build? ?ProjectFile?
```

If you are building from the command-line, it is generally recommended that `--build` be the last option before the project file name. It tells InstallJammer where the options and switches end and where the project file name begins. Every argument after the `--build` will be joined together to make the file name, so you can include spaces on the command-line, and InstallJammer will join them properly.

The same is true for other options. Anytime an argument is encountered that doesn't match one of the given switches or options, its value will be appended to the value of the previous option. This allows for spaces on the command-line, and InstallJammer will join them properly where it makes the most sense.

Example:

```
installjammer -DAppName My Application --quick-build --build /path/to/my/project file.mpi
```

The arguments of this command-line will be joined like this:

```
-DAppName "My Application" --quick-build --build "/path/to/my/project file.mpi"
```

Command-line Options

`--build`

This option specifies that the given project file should be loaded and built without loading the install builder GUI. This option is good for building installs from other programs or at regular times using scheduling. This should generally be the last switch specified before the project file name.

`--build-dir <directory>`

Specifies a directory to use as the temporary directory when building. InstallJammer will attempt to create the directory if it does not already exist.

`--build-for-release`

If this option is specified, the project file being built will be built for final release.

`--build-log-file`

Specifies a filename to save the build log information to. The default is `build.log` in the project directory.

`--control-script <scriptFile>`

This option specifies a script to source in when InstallJammer is building from the command line.

`-D<option> <value>`

Set the value of option to value. Example:

```
-DAppName "My Software"
```

Will override the default value stored in AppName with the value My Software. This is useful for over-setting options in different installs during command-line builds so that a single project file can be manipulated to be many different builds.

Any number of options can be specified in this manner. Each one will be processed in the order in which it was specified on the command line.

--help

Displays usage information for InstallJammer.

--output-dir <directory>

Specifies the output directory to store the resulting installers in when they are built. InstallJammer will attempt to create the directory if it does not already exist.

--platform <platform>

This option can be specified with the -b option to tell InstallJammer to only build certain platforms. Each platform name specified with a -p will be built instead of building all of the platforms (the default option). More than one -p platform can be specified, and each platform will be appended to the build list.

--quick-build

If this option is specified, the given project file will be rebuilt without repackaging the files for the installer just as a quick build in the GUI.

--quiet

Turn off messages from InstallJammer while building. Errors will still be reported on stderr.

--test

This option can be specified with the --build option to tell InstallJammer to test the install once it has been built. The install builder GUI will still not be loaded, and the install will execute when the build is complete.

--verbose

This option tells InstallJammer to show verbose output when building.

--version

Display version information for InstallJammer and exit.

-0-

General Information

Application Information

Application Information

Application ID

A unique identifier for this application.

The Application ID is defined when the project is first created and is used to identify the application on subsequent installs to the same machine and during uninstallation. The Application ID can be accessed with the `<%ApplicationID%>` virtual text.

Application Name

The name of the application you are installing.

This name will be used throughout the install when the application name is used. It can be accessed by using the virtual text `<%AppName%>`.

Application URL

This is the main web address for information about your application. This is used for information to tell users where they can find your application on the web. The application URL can be accessed by using the `<%ApplicationURL%>` virtual text.

Company

Your company name.

This field can be used throughout the install as an easy way to retrieve the name of the company producing the application. By default, InstallJammer uses the company in its default install location. The company name can be accessed by using the virtual text `<%Company%>`.

Copyright

This property specifies your copyright notice, if any. This will be included in the Copyright field in the resources of a Windows executable. The copyright can be accessed by using the `<%Copyright%>` virtual text.

Install Icon

The icon used throughout the install. If the icon path given is a relative path, InstallJammer will first attempt to locate the image file in the project's own directory and then in InstallJammer's root directory.

By default, all panes within the install that contain an icon use this icon, but most install themes will allow you to change the icon for each pane. If the pane is not modified, this icon will be used when applicable. The install icon can be accessed by using the virtual text `<%Icon%>`.

Install Image

The main image used throughout the install. If the image path given is a relative path, InstallJammer will first attempt to locate the image file in the project's own directory and then in InstallJammer's root directory.

By default, all panes within the install that contain an image use this image, but most install themes will allow you to change the image for each pane. If the pane is not modified, this image will be used when applicable. The install image can be accessed by using the virtual text `<%Image%>`.

Install Version

A numeric version used by InstallJammer for your application.

Install versions are used during installation to determine what version files within the install are. This is useful for when you want to overwrite certain files that may already exist but only if the install has a higher version. The version can be accessed by using the virtual text `<%InstallVersion%>`.

Short Application Name

A shortened name for your application. This is usually all lowercase and contains no spaces and is used on UNIX systems in the installation directory. Even though most all modern UNIX systems support spaces in file and directory names, it's usually considered bad form. The short application name can be accessed by using the virtual text `<%ShortAppName%>`.

Upgrade Application ID

If this property is not null, it specifies the Application ID of another install project that this project is meant to upgrade. An upgrade installer is one that skips some of the procedures of a regular installer and adds its information to the uninstall of the previously-installed application. This upgrade application ID can be accessed by using the `<%UpgradeApplicationID%>` virtual text.

Version String

The version of the application.

The version string is for you to identify the release version of your application. InstallJammer does not use this version for anything, so it can be anything you like. The version string can be accessed by using the virtual text `<%Version%>`.

Install Features

Allow Language Selection

If this property is true, the user will be allowed to select the language to use for messages during installation. If it is false, the Default Language is always used.

Cancelled Install Action

What InstallJammer should do if an install is cancelled.

If the user cancels an install after the main installation of files has already begun, some of the application is already copied onto the system. This option lets the installer rollback any modifications that have been made by the installer since it started running.

Default Language

This property specifies the default language to use when installing. InstallJammer will attempt to determine the best language based on the user's system settings, but if no settings can be found, this language will be the default in the language selection. If no language selection is allowed, this default language will always be used.

Default to System Language

If this property is true, the installer will use the default language on the target system if that language has been compiled into the installer. If the system language is not part of the installer or if this property is false, the Default Language will be used.

Enable Response File

If this property is true it will enable the `--response-file` and `--save-response-file` command-line options in your installer. This allows a user to pass `--save-response-file` when running an installer and save their responses to an output file that can then be read back in during later installs using the `--response-file` option. This is very handy for users who need to run an installer on multiple machines or for multiple people where it should all be run the same way. Read more about what virtual text is included in a response file in the [Add](#)

[Response File Info](#) action.

Extract Solid Archives on Startup

If this property is true, and the installer includes any solid archives, they will be extracted as soon as the installer starts up. The user will be shown a progress bar as the archives are extracted. This option is only useful if you need your files extracted before you begin the actual installation process. By default, InstallJammer will extract the solid archives just before it begins the actual file installation.

Wizard Height

Specifies the height of the installer and uninstaller wizard window. Defaults to whatever is the default for the chosen installer theme.

Wizard Width

Specifies the width of the installer and uninstaller wizard window. Defaults to whatever is the default for the chosen installer theme.

Project Preferences

Command Line Failure Action

What InstallJammer should do if a file is missing from an install when building from the command-line.

When InstallJammer is building from the command-line (by passing a -b option), the user doesn't have the ability to recognize problems and correct them. This option specifically deals with the case that a file that should be part of the install is missing, and InstallJammer needs to know what to do.

Compression Level

Define the compression level to use when packing files.

This tells InstallJammer what level of compression to use when packing files in an installer. 1 means lower compression levels but a faster rate of packing and unpacking. 9 means higher compression but sometimes slower unpacking. The default level is 6.

Compression Method

This specifies the default compression method to use when packing files into the installer. The choices are:

- lzma
- lzma (solid)
- none
- zlib
- zlib (solid)

A solid archive is when all of the files to be packaged are actually packed into a single file before they are compressed into the installer. This has the advantage of a much higher rate of compression since the entire file is compressed at once instead of a bunch of little files compressed separately.

Solid archives cannot be read directly from within the installer though, so they must first be extracted to a temporary location before they can be installed. This will require the target system to have a temporary location that is big enough to hold your entire, uncompressed archive before installation begins.



ZLIB is the default compression method because it is fast, efficient, and it doesn't require a separate extraction step.

LZMA can be a much better compression method in some cases, but it is highly recommended that you not use lzma compression unless it is a solid archive. LZMA compression is very slow and doesn't gain much on most files over ZLIB, but it can be a big win when using solid archives.

If you have a larger file that would benefit from LZMA compression, you can always compress just that one file by specifying the Compression Method in the file group tree. InstallJammer can easily compress every single file in an installer with a different compression method if you want.

Default Directory Location

If this property is set, it specifies the default Location to use for all files and directories that are direct descendants of a file group if their own Location property is not set. This allows a project to specify that every file or directory in a project should be located in a particular place on the system or relative to the project file, for example. If a file or directory specifies its own Location, it overrides this default value.

Include Install Debugging Options

Include command-line debugging options in the install.

During testing, InstallJammer includes a small set of debugging options that can be passed to installs on the command-line. These options allow the install builder to debug their install while it is running.



Debugging options should be turned off before distributing a release of your installer. This can be done by disabling this option or by building your installs for final release.

Preserve UNIX File Permissions

If this property is true, InstallJammer will record the current file permissions on each file and directory stored in the installer and restore them when they are installed. Any permissions which have been set for a file or directory in the project will override these permissions, but by default, the permissions recorded on the build system will be restored on the target install system.

Preserve Windows File Attributes

If this property is true, InstallJammer will remember the file attributes on each file and directory stored in the installer and restore them when they are installed. Any attributes which have been set for a file or directory in the project will override these attributes, but by default, the attributes recorded on the build system will be restored on the target install system.

Refresh File List Before Building

If this property is true, InstallJammer will automatically search for new and deleted files in directories before building any installers. This will automatically pick up any changes made to the directories in your installer before the new installers are built.

Save Only Toplevel Directories

This option tells InstallJammer that you only want to save the files and directories that are a direct child of a file group. This means that directories in a file group will not be saved recursively. This option is useful for building projects dynamically using command-line tools where you don't want the GUI builder to save the files for you.

File Groups and Directories in a project have a Save Files property which basically acts as a finer-grained version of this property. You can set that a particular File Group or directory should not save its files instead of the entire project.



Saving only toplevel directories should be used with caution. Any change you make to any file or directory that is not a toplevel will be lost since it will not be saved in the

install project.

Skip Unused File Groups

If this property is true, it means that InstallJammer will skip any file groups it finds that are not being used by a component. This can be an easy way of disabling an entire file group without removing it from the project. If this property is false, all file groups will be included in the installer regardless of whether they are used.

-0-

Platform Information

Platform options describe options that are specific to each platform in an install project. When an install is built, these options are saved only in the installer for that platform.

Active

Whether or not this platform is active.

When a platform is inactive, nothing in InstallJammer will use it or affect it. Mainly this means that an installer for the inactive platform will not be built, but several other things are also disabled when a platform is inactive. All of the properties for the platform will still be saved with your project, so a platform can easily be enabled again at a later time.

You should mark a platform as inactive if you don't plan to distribute your application for that platform.

Build Separate Archives

If this property is true, the installer for this platform will be built in separate archive files instead of being built as a single executable file. This is useful for splitting up larger files for when installation will be done from a CD or DVD.

Default Destination Directory

The default destination directory for your application.

This is where your application will default to installing on the target system if the user does not modify the directory (or is not given the option to modify the directory).

Default Directory Permission Mask (UNIX only)

The default permission mask to use when installing directories.

When a directory is created on the target system during installation, this option tells InstallJammer the default permission mask to use when creating. If a separate permission has been set for the directory, it will override the default permission.

Default File Permission Mask (UNIX only)

The default permission mask to use when installing files.

When a file is installed on the target system during installation, this option tells InstallJammer the default permission mask to use when installing. If a separate permission has been set for the file, it will override the default permission.

Default Install Mode

This property sets the default install mode for the installer if the user has not specified a different mode on the command-line or if the user is not given the option to specify the mode. This allows you to make a console or silent only installer that the user cannot use any other way.

Default Program Folder (Windows only)

The default Program Folder name under the user's menu.

This is the default Program Folder to store your application's shortcuts in on the Windows platform. If you create shortcuts in the <%ProgramFolder%> directory, this is the default name for that folder.

Default Setup Type

The default setup type chosen during installation.

If the user is allowed to change the setup type during installation, this will be the default one

chosen. If the user is not allowed to change the setup type, this will be the setup type that is installed by default.

Fall Back to Console Mode (UNIX only)

This property tells InstallJammer that the (un)installer should fall back to the console mode of installation if the GUI mode fails to initialize for some reason. This means that if the user chooses a GUI mode (or is defaulted to one), but they do not have a GUI environment, the installer will automatically detect and launch into the console install.

Include Windows API Extension (Windows only)

Whether or not to pack the TWAPI extension into your installer.

The Tcl Windows API Extension provides commands at the Tcl scripting level to some of the internal Windows APIs. This provides the ability for you to script Windows-specific features into your installer (like services, user account commands, etc...) that would not normally be available.

Install Executable Description (Windows only)

This property describes the text to be used in the File Description field of a Windows install executable. The default is "<%AppName%> <%Version%> Setup".

Install Executable Name

The name of your installer executable.

This is the file that InstallJammer will create when it builds your installer. The installer will be built in the build subdirectory of your project directory.

Install Program Folder for All Users (Windows only)

If this option is true, InstallJammer will create the Program Folder under the Start menu for all users of the system and not just the user installing the application.

Program Executable

The main executable for your application.

This field contains the main executable for your application. It is used throughout the install for things like creating shortcuts to your application and starting your application when the install is complete.

If your program does not have an actual program entry point, you can just leave this blank.

Program License

The LICENSE file for your application.

Your application is not required to contain a LICENSE file, but some do. If you have one, this value tells InstallJammer where to look for it when offering the user the ability to read it.

It is also appropriate, in most cases, to create a shortcut to your LICENSE file in your Program Folder.

Program Readme

The README file for your application.

Your application is not required to contain a README file, but some do. If you have one, this value tells InstallJammer where to look for it when offering the user the ability to read it.

It is also appropriate, in most cases, to create a shortcut to your README file in your Program Folder.

Prompt for Root Password

If this property is true, and Require Root User is true, the user will be asked to specify their

root password to continue instead of just getting a failure message and the installer exiting. When the user enters their root password, the installer will be restarted with the same options.



InstallJammer attempts to use the given tools for the current desktop environment if available. KDE will use kdesu. Gnome will use gksu or gnomesu if found. All other environments will attempt to use one of the previously-mentioned tools if found. Finally, if none of these tools can be found, the user will either get an xterm if they are not running from a console, or it will simply ask for the password in the console if they are.

Require Administrator (Windows only)

If this property is true, the resulting installer will only work if the user is an administrator on the target Windows system. For Vista and beyond, this means the installer is built as an executable that requires administrator privileges, and the user will be asked for an administrator password before the install is executed. On earlier versions of Windows, the installer will check to see if the current user is in the Administrators group and will show an error message and exit if the user is not an administrator.

Require Root User (UNIX only)

Whether or not the user must install this application as root.

Some applications require a root user to install on a UNIX system. This option tells InstallJammer that the user cannot run the install unless it is run as root. InstallJammer will check the username during install initialization and fail with an error message if the user is not root.

Root Destination Directory (UNIX only)

The default destination directory when a user is installing as root.

In most cases, an application can either install in a system location if the user is root or into the user's home directory if the user is not root. The root destination directory tells InstallJammer where the application should be installed if the user is installing as root. If the user is not root, or this option is left blank, InstallJammer will fall back to the Default Destination Directory for all installs.



This option does not need Require Root User. The default destination will be adjusted if the user is root even if you don't require them to be root (unless it is blank).

Use Uncompressed Binaries (Windows only)

If this property is true, the Windows installers will be built as uncompressed binaries. Installers built with InstallJammer are compressed using varying compression technologies for each platform. This is an attempt to keep the installers smaller at the cost of a slight increase in startup time. Some anti-virus software can mistakenly mark a compressed binary as having a virus when they actually don't. For this reason, it is sometimes best to build with the original, uncompressed binaries on Windows platforms. If you do build with uncompressed binaries, your installers will be several megabytes larger than the compressed versions.

Windows File Icon (Windows only)

The .ico to associate with your installer. If the icon path given is a relative path, InstallJammer will first attempt to locate the .ico file in the project's own directory and then in InstallJammer's root directory.

If specified, the icon shown on Windows in the Explorer and as the main icon in the title bar of your installer will use the values in this file. You must specify an icon file that contains all of the possible size / color combinations for an icon.

A default icon is associated with installers on Windows, and some icons have been provided that meet the specifications.

-0-

Package and Archive Information

Package Information

Package information can be used by you in your own installer, but its main use is by the actions in InstallJammer that register your application with the different packaging systems. Since most of the information is the same across package databases, this section provides an easy way to specify the information once and then let InstallJammer use the right bits for each different package system.



This information is used in the [Register Package](#) action. If you plan on registering your application with the system package manager, it is highly recommended that you fill in all the information for your package. Some package managers will not accept a package that is missing pieces of the following data.

Package Description

This is a long text description of your package. This can include multiple lines that describe what your application is and what it does.

Package License

This should be a one line description of the license your application uses. This is usually something very simple like: GPL, MPL, Apache, etc...

Package Maintainer

This is a name and e-mail address of the person who maintains your application should you want to be contacted. This is in the form of: Person's Name <email@address.com>

Package Name

The name your application will have in the package database. This should be all lowercase and short. It should not be the full name of your application.

Package Packager

The name and/or e-mail of the person who packages your application. This would be you, the person building the installer.

Package Release

The release version of your application. By default, InstallJammer uses the <%PatchVersion%> from your <%InstallVersion%> for this value.

Package Summary

This is a one line summary description of your package. This is the short description that briefly describes your package. The Package Description gives the long version.

Package Version

The version of your application. InstallJammer uses <%MajorVersion%>.<%MinorVersion%> from your <%InstallVersion%> by default since that's what most package managers expect.

Tar Archive Information

This section describes the information for building a tar file of your application. If this section is active, InstallJammer will build a tar file alongside your other platform-specific installers.

Active

If true, InstallJammer will build a tar file of your application when it builds the rest of your installers.

Compression Level

The level of gzip compression to use when compressing your tar file. If this is 0, the file will

not be gzipped at all and will just be a plain tar file. You will want to remove the .gz from the output file name if you are not using compression.

Default Directory Permission Mask

The default permission mask to use when storing directories. If a separate permission has been set for the directory, it will override the default permission.

Default File Permission Mask

The default permission mask to use when storing files. If a separate permission has been set for the file, it will override the default permission.

Output File Name

The name of the tar file to output. This file will be in the output/ subdirectory of your project directory alongside your other installers.

Virtual Text Map

This is a map of strings to use on each file and directory stored in the tar file. A string map is a list of pairs that specify that one string should be mapped to another. The default is to map all instances of <%InstallDir%> in the destination file name to <%ShortAppName%>. That means that all files being installed in <%InstallDir%> will end up in a subdirectory after your short application name with the rest of their directory structure intact.

Zip Archive Information

This section describes the information for building a zip file of your application. If this section is active, InstallJammer will build a zip file alongside your other platform-specific installers.

Active

If true, InstallJammer will build a zip file of your application when it builds the rest of your installers.

Compression Level

The level of compression to use when compressing your zip file.

Output File Name

The name of the zip file to output. This file will be in the output/ subdirectory of your project directory alongside your other installers.

Virtual Text Map

This is a map of strings to use on each file and directory stored in the zip file. A string map is a list of pairs that specify that one string should be mapped to another. The default is to map all instances of <%InstallDir%> in the destination file name to <%ShortAppName%>. That means that all files being installed in <%InstallDir%> will end up in a subdirectory after your short application name with the rest of their directory structure intact.

-o-

Components and Files

Groups and Files

The file groups tree is used to coordinate the files and directories that will be included in your installer. Files and directories are stored under a file group, and file groups are then made part of a component (or multiple components).

When an install is first created, a single file group called "Program Files" is created with the project. If you are only creating a simple installer for your application, this one file group is usually all that you need.

See [Files and Directories](#) for more information about the files and directories that are contained within file groups.

File Group Properties

File groups are further defined by their individual properties. When a file group is created, it is setup with a default set of properties that you can alter if you need to. Each file group has a set of standard properties that are properties that are shared by many components in InstallJammer and a set of Advanced Properties that are specific to file groups.

Standard Properties

See [Standard Properties](#).

Active

If a file group is marked inactive, all the files and directories included in that group will not be included in the installer when it is built.

Compression Method

This property defines the method of compression for the given file group. All other files and directories in the given file group will have the same compression method. If the compression method is left blank, the default compression method for the project will be used.

Destination Directory

The directory this file group will be installed to on the target system. The default is <%InstallDir%>, the main installation directory.

Display Name

If this property is set, it specifies the name to display when the name of the file group is shown during installation. If this property is left blank, the actual name of the file group will be used.

File Update Method

This property tells InstallJammer what to do with files when it is updating them on the target system. Updating is when the file already exists, and we need to determine what to do with our file.

Follow File Links

If this property is true, any linked files found within the file group will be followed and the file they link to will be stored as an actual file in the installer. If this property is false, any links will be saved as links in the installer and recreated as a link on the target system during installation.

Name

The name of this file group.

Save Files

If this property is false, only the toplevel directories in the file group will be saved. All of the files and subdirectories of the top directories will not be stored in the project file and will be discovered at build time. This is useful when you only want to store a directory but want everything in that directory to be built dynamically.

Size

If the size property is specified, it tells InstallJammer to override what may actually be the size of the files stored in the file group and use the given size instead. The size should be given as an integer only which will automatically be converted to a readable size during installation.

Version

This property tells InstallJammer what Install Version to store for this file group on the target system. If this property is left blank, InstallJammer will use the global `<%InstallVersion%>` variable for the project.

Build Platforms

This is a list of all the platforms that are supported by InstallJammer and whether or not this file group should be built for each platform. If a platform is marked No, this file group will not be included when the installer for that platform is built.

It is possible to build a file group into multiple platforms and then not install the file group based on parameters during installation, but this option specifically tells InstallJammer to not even include this file group for the given platform.

Permissions

This feature allows you to set the file permissions for the given file group, file or directory. By default, no permissions are set which means that the file or directory will be installed with some default permissions on the target system.

Permissions on a file group refer to the directory that will be created for the file group and not to all of the files in the file group. Directories and files can be set with their own permissions or be set as group when multiple files and directories are selected.

Both Windows attributes and UNIX permissions are provided and will be applied based on the target system. So, if a file has both Windows attributes and UNIX permissions, the Windows attributes will be applied when the file is installed on a Windows system, and the UNIX permissions will be applied when installed on a UNIX system.

Creating a new file group

Click the Add New File Group button on the toolbar to add a new file group to your install.

A new file group is created with default values that you can change after its creation. Your new file group will be at the bottom of the list.

Deleting a file group

Click the Delete button on the toolbar to delete the selected nodes from your install. This can also be done through the Delete key on your keyboard.

Any selected nodes in the file group tree will be deleted. This includes files, directories and file groups.

Adding files to a file group

Click the Add Files to File Group button on the toolbar to add new files to the selected file

group. A file dialog will open and ask you to select the files you wish to add to the file group.

Files can also be added to a program folder by dragging and dropping them from your file explorer. Most major window managers are supported.

Adding directories to a file group

Click the Add Directory to File Group button on the toolbar to add new files to the selected file group. A directory selection dialog will open and ask you to select a directory you wish to add to the file group.

Directories can also be added to a program folder by dragging and dropping them from your file explorer. Most major window managers are supported.

Deleting files from a file group

Click the Delete button on the toolbar to delete the selected nodes from your install. This can also be done through the Delete key on your keyboard.

Selected nodes in the file group tree, including files and directories directly beneath the file group, will be deleted from the project.

Only files and directories that are directly beneath a file group can be deleted. Subdirectories and files that are contained within directories cannot be deleted from a project. If you do not wish to include files or directories that are contained in a subdirectory, you simply uncheck them. Files and directories that are unchecked are not included in an install when it is built.

This is done so that once you include a directory in a file group, InstallJammer will maintain the entire directory structure regardless of what is included in the installer. You can always add other subdirectories directly to the file group and have them install anywhere you like based on Destination Directory.

Excluding files and directories from a project

Files and directories can be excluded from a build by unchecking (or de-activating) them in the file group tree. Any file or directory that does not have a check mark is considered inactive and will be ignored by InstallJammer when building your installer.

Disabling a directory will automatically disable everything contained in that directory, even if other files and directories in the directory are checked. Once you disable a directory, all of its subdirectories and files are disabled as well. If you wish to disable an entire directory but still include subdirectories beneath it, you will need to add those subdirectories directly to the file group. You can modify their install locations through the Destination Directory property.

The File Explorer

The File Explorer button will bring up a directory and file explorer which can be used to drag-and-drop files and directories into file groups. If you did not specify a file explorer in your preferences, InstallJammer will attempt to find the best one based on your window manager.

On Windows, this defaults to Windows Explorer. You can specify exactly which file explorer you want to use in your preferences. InstallJammer supports drag-and-drop on most major window managers.

Files and Directories

Files and directories are added to your project in File Groups. Each file or directory in your project has its own set of properties that will override the default values specified in its File Group. By default, a file or directory will inherit the properties of its parents up the tree, going back as far as the File Group.

When adding files and directories to a file group, they begin with no specific options, which means that they will inherit the properties of the File Group. As you modify the properties of a directory, it will automatically affect the properties of all files and subdirectories in that directory. Modifying the properties of a file will only change that file. You can also select multiple files and directories and change their properties all at once.



It is important to remember that properties on file groups and directories propagate down and affect all files and directories beneath them. Looking at the File or Directory Details will show you what values currently affect the selected file or directory.

Files that have been newly-added to the project since the last time the project was loaded will appear in blue. This makes it easy to detect when InstallJammer has modified the current project because it has automatically picked up a new file while recursively searching through directories.

Files that are disabled or "greyed-out" mean that InstallJammer cannot find the file on the local system. The file may have been deleted on the system, or the location of the file may have changed. You can find the file again by changing its Location property or by changing the Location of its containing directory. Files that cannot be found will not be included when the install is built.



Files cannot be deleted from an included directory in the file tree. InstallJammer keeps track of the entire structure of a directory, so deleting a file from the project would only result in InstallJammer finding it again the next time you build. If you don't want to include a file in your installer, just uncheck it or mark it inactive.

Standard Properties

See [Standard Properties](#).

Active

If a file is marked inactive, that file will not be included when the installer is built. If a directory is marked inactive, that directory and all files and directories beneath it will not be included when the installer is built.

Compression Method

This property defines the method of compression for the selected file or directory. If specified on a directory, all other files and directories beneath the selected directory will have the same compression method. If the compression method is left blank, the default compression method for the project will be used.

Destination Directory

This specifies the directory on the target system that this directory or file will install into. Modifying the Destination Directory of a directory will automatically affect the destination of every file and directory in that directory since they inherit properties from their parent.



If the Destination Directory is set on a directory, it means the name of the directory where this directory will be created on the target system. The Destination Directory for a file is the directory the target file will be placed INTO on the target system. See the Install Location under File or Directory Details to know for sure where a given file or directory

will be installed.

File Update Method

This property tells InstallJammer how to behave when it is installing a file or directory on the target system that already exists. Modifying the File Update Method of a directory will automatically affect the update method of every file and directory in that directory. This does not modify their File Update Method property, but they will inherit from their parent.

Location

This property specifies the location of the given file or directory on the local system. When changing the location of a file or directory, you can specify that the location provided should be stored in the project file itself or in your local preferences. If the location is stored in your local preferences, it will not affect the location in the project file and will only be used when the project is loaded on the current machine. This allows developers to specify different locations for projects on their own machines without having to change the locations within the project file itself.

If no location is specified, the location of a file or directory will be derived from its parents going up the tree. Looking at the File Details for a particular file or directory will show you the location where InstallJammer has found it.

Target Filename

This file-only property specifies the name that this file should be created as on the target system during installation. The Destination Directory can specify where this file should be installed, but this property specifies what it will be named when it gets there.

Version

This property tells InstallJammer what Install Version to store for this file or directory on the target system. Modifying the Version of a directory will automatically affect the version of every file and directory in that directory. This does not modify their Version property, but they will inherit from their parent.

File and Directory Details

This property block will tell you all of the information about the selected file or directory based on its own properties and the properties it has inherited from its parents. This read-only block contains mostly information from the build system about the file or directory, but a few properties are worth noting.

Compression Method

This is the method that this file will be compressed with. This is derived from its parents or from the default compression method for the project.

File Update Method

This property tells you how the selected file or directory will behave during installation if the target already exists on the target system. If the selected object has no File Update Method of its own, it will inherit up the tree until it finds one.

Install Location

This property tells you how where the selected file or directory will be installed on the target system. If the selected object has no Destination Directory of its own, it will inherit up the tree until it finds one.

Location

This is the location on the local system where the file or directory resides. This can be specified in the Location property, or it will be derived from the parents going up the tree.

Version

This property tells you the Install Version of the selected file or directory when it is installed.

This version will be stored for future installations after the install is complete. If the selected object has no Version of its own, it will inherit up the tree until it finds one. If no Version is found in any parent, the global <%InstallVersion%> for the project will be used.

Permissions

This feature allows you to set the file permissions for the given file or directory. By default, no permissions are set which means that the file or directory will be installed with some default permissions on the target system.

Permissions for a directory are not inherited by the files and directories underneath it. The permissions only apply to that directory. Multiple files or directories can be selected and their permissions set all at the same time.

Both Windows attributes and UNIX permissions are provided and will be applied based on the target system. So, if a file has both Windows attributes and UNIX permissions, the Windows attributes will be applied when the file is installed on a Windows system, and the UNIX permissions will be applied when installed on a UNIX system.

Save Files

If this property is false, all of the files and subdirectories in this directory will not be saved in the project file. When this directory is built, all of its contents will be dynamically generated from whatever is currently in the directory. This allows for adding directories where all of the contents are only stored at build time. This property works like a finer-grained version of the Save Only Toplevel Directories project preference.

-0-

Components

Components are used to group file groups in your install into user-friendly categories. Each component can contain any of the file groups in your project, and if the user is presented with the option, they can choose which components they wish to install.

By default, a single component called Default Component is created for your project, and it contains the Program Files file group.

Standard Properties

See [Standard Properties](#).

Checked

If this property is true, this component will be displayed as checked when it is first added to a component tree. This will automatically set the active status of the component depending on the value when it is first added to a tree.

Component Group

If this property is specified, it tells InstallJammer to put this component into a specific group with other components. Each component that shares the exact same name of group will be grouped together as radio buttons instead of check buttons. You must make sure that each component you want in a radio group shares the same exact Component Group name.

Name

The name of this component.

Required Component

This option tells InstallJammer that this component is required for all installations and will therefore not be available for the user to uncheck during installation. The component will still appear in the list of available components, but it will be disabled for the user.

Selectable

If this property is set to No, the component is displayed in a component tree as a header with no check or radio button.

Show Component

If this property is false, the component will not be added to a component tree when it is drawn. This can be used to hide components that you don't want shown during installation.

Size

If the size property is specified, it tells InstallJammer to override what may actually be the size of the files stored in the component and use the given size instead. The size should be given as an integer only which will automatically be converted to a readable size during installation.

Text Properties

Description

This is a brief description of the contents of this component. If the user is able to choose which components they wish to install (during a Custom installation) they will see this description for each component to tell them what is in it.

Display Name

The name of the component to display when populating the components on a pane. If this property is left blank for a given language, the installer will use the standard Name property instead.

Build Platforms

This is a list of all the platforms that are supported by InstallJammer and whether or not this component should be built for each platform. If a platform is marked No, this component will not be included when the installer for that platform is built.

It is possible to build a component into multiple platforms and then not install the component based on parameters during installation, but this option specifically tells InstallJammer to not even include this component for the given platform.

Creating a new component

Click the Add New Component button on the toolbar to add a new component to your install.

A new component is created with default values that you can change after its creation. Since components can be nested, your new component will be a child of whatever component is currently selected when you create your new component.

Deleting a component

Select the component you wish to delete and click the Delete Component button on the toolbar or the delete key on your keyboard.

-0-

Setup Types

Setup types define a method of installation when the install is run. The user can choose the method of installation based on the setup types available. Each setup type has a list of included components that are installed if the user chooses that setup type.

The "Custom" setup type has special meaning to InstallJammer. All other setup types are regarded as normal and have no special features. If the user is offered the ability to choose their setup type, and they choose Custom, they should also be offered the ability to select the individual components they wish to install. Any component added to the Custom setup type will be available to the user to choose during installation.



By default, InstallJammer creates two setup types: Typical and Custom.

Standard Properties

See [Standard Properties](#).

Name

The name of this setup type.

Show Setup Type

If this property is false, this setup type will not be shown when populating setup types in a pane. This is used to create setup types that you want hidden from the user during installation.

Text Properties

Description

This is a brief description of the contents of this setup type. If the user is able to choose which setup type they wish to install they will see this description for each type to tell them what is in it.

Display Name

The name of the setup type to display when populating the setup types on a pane. If this property is left blank for a given language, the installer will use the standard Name property instead.

Build Platforms

This is a list of all the platforms that are supported by InstallJammer and whether or not this setup type should be built for each platform. If a platform is marked No, this setup type will not be included when the installer for that platform is built.

It is possible to build a setup type into multiple platforms and then not install the setup type based on parameters during installation, but this option specifically tells InstallJammer to not even include this setup type for the given platform.

Creating a new setup type

Click the Add New Setup Type button on the toolbar to add a new setup type to your install.

A new setup type is created with default values that you can change after its creation.

Deleting a setup type

Select the setup type you wish to delete and click the Delete Setup Type button on the toolbar

or the delete key on your keyboard.

-0-

User Interface and Events

Panes and Actions

The Install / Uninstall Panes and Actions section of the install builder shows a tree of your install's user interface. The user interface is first defined by a set of possible install types. Underneath that is a list of all of the panes that are included in that install type. Each pane can have any number of actions that execute as the user moves through the install.

Panes

Selecting a pane in the tree will bring up some information for that pane as well as properties that you can modify to customize the pane. See [What are Panes?](#) for more details.

Actions

Selecting an action in the tree will bring up some information for that action as well as properties that you can modify to customize the action. See [What are Actions?](#) for more details.

The user interface tree defines how the user will see your install when they run it. It contains several install modes that are defined by InstallJammer, and beneath those are the panes that are included for that install mode. Each pane can have any number of actions attached to it.

The User Interface Tree



Panes and actions that have been added since the project was last loaded will appear in blue. A * beside a pane or action's title tells you that the given pane or action has conditions attached to it.

Common Components

This section of the interface tree is used to hold panes of an install theme that don't belong anywhere in the install process itself. Panes installed in the Common Components section can be modified just like normal panes, but you don't have the option to add the same pane to an install type.

Common examples of this are a background window that covers the screen behind the install wizard with some text or images.

Standard Install

This section contains the panes used by InstallJammer for a standard installation. A standard installation is when the user chooses to install the application just as you built it. All of the panes will be shown in order as they are shown on the interface tree.

Default Install

A default install is when the user has chosen to accept all of your default settings and just do the install without changing anything. The default install usually contains just a few screens to prompt the user to confirm installation and then to copy all of the files.

Console Install

A console install is when the user wants to install the program from a command line instead of through the GUI interface. A console install will run inside a terminal window and everything will be input and output to and from the terminal.



Console installs are not supported on Windows.

Silent Install

A silent install is when the user wants to accept all of the default options and install the program silently without a GUI interface. This is useful for some administrators who have to install an application on many different machines.

A silent install does not have any panes, only actions. Because of this, you will only see a list of actions under the Silent Install, and you are not allowed to add panes.

Action Groups

Action groups are a way to group actions together into one place. An action group can contain any number of actions to execute and has its own conditions and properties as well as those of the actions it holds.

Action groups are best used if you define an alias for them so that you don't always have to remember the ID of the action group. This is how InstallJammer sets up the default Startup Actions and Install Actions groups when it creates a new project. You can execute all of the actions in an action group by using the Execute Actions action.

See [What are Action Groups?](#)

Previewing a pane

Once you've selected a pane you will see a button in the bottom right-hand corner of the properties window to preview the pane. Clicking this button will pop up a sample wizard with the pane you're currently viewing. This lets you see the pane the way it will appear during installation.

Most options, if changed, will automatically affect the appearance of the preview window, so you should be able to make textual and design changes, click Preview Pane again, and immediately see your changes. Not all changes are immediately seen. Some changes and options can only be seen when actually running the install. In this case, simply build and run a test install of your project to see what the panes will look like.

Editing pane code

Each pane in an install is defined by the Tcl/Tk code that built it. InstallJammer gives you the freedom and flexibility to modify each pane in your install to be exactly what you want it to be. Simply click the Edit Pane Code button when editing a pane, and an editor will pop up with all of the Tcl/Tk code used to create the pane you're looking at. If you don't understand Tcl/Tk code, you might want to read [What is Tcl/Tk?](#) as a starting point.

Restoring original pane code

The Restore Original Pane button will restore the original code for the pane currently being edited. This is useful if you wish to remove the changes you've made to the pane or you accidentally messed up the pane code. Restoring the pane code will put the pane code back into its original state. It will not modify your properties though. Only the code of the pane.

-0-

Command Line Options

The command line options panel allows you to specify the different command-line options that will be available in your (un)installer. InstallJammer sets up some default options for each new projects, but these can all be removed if you want.



There are a few options that are defined internally during startup that cannot be overridden by your options. They are:

- help, which displays the usage information for the (un)installer
- temp, which allows the user to specify a temporary directory
- version, which shows version information for the (un)installer

If response files are enabled, the following options are also added to each installer:

- response-file, which specifies a response file to read in
- save-response-file, which specifies that responses should be saved to the given file

InstallJammer automatically defines the character that precedes an option depending on platform, and this cannot be changed. The characters are removed from the option before processing, so options should be specified without any preceding characters.



A / is used as the option character on Windows, and -- is used on UNIX, but it really doesn't matter. InstallJammer will accept either one on either platform, but it will always display the correct one when showing usage information.

Command Line Option Properties

Option

The name of the option not including any preceding option characters. Options are case-insensitive when specified on the command-line by the user, but they will maintain case when displaying them in the usage information.

Virtual Text

The name of the virtual text (without <% and %>) to store the result of the option into. What is stored in the virtual text result depends on what type of option is being used.

Type

Each option is of a specific type, and each type behaves differently. The types are:

Boolean

A boolean option means that the user must specify Yes or No or some equivalent as the argument to the option. The virtual text will be set to 0 if the user specified No or 1 if they specified Yes.

Choice

A choice is an option that has a list of possible choices the user can choose from. The choices are all case-insensitive, but the virtual text will be set case-sensitive. For example, if you had the option *check* that accepted *Foo* or *Bar*, the user could pass */check foo* on the command-line, but your virtual text would still be set to *Foo* even though the user passed an all lowercase version.

Prefix

A prefix option means that the given option is only a prefix that can be specified along with any option after the prefix and a given value. This can be used to provide a generic option for specifying many things in your installer. Any option provided to a prefix switch is set as virtual text with the given value. If the Virtual Text property is not null, it is

prepended to the option given.

For example, if you had the prefix option D with Virtual Text CommandLine, the user could pass:

/DFoo bar

On the command-line, and you would have the virtual text <%CommandLineFoo%> set to bar.

String

A string option means that the user can specify anything they want and no checks are done for validity. You will have to check the value yourself using an action. The virtual text will be set to the value of the given string.

Switch

A switch means that the option is specified with no argument and simply turns on the option. The virtual text will be set to 1 if this option is passed on the command-line or not set at all if it is not passed.

Debug

This tells InstallJammer whether the given command-line option is meant to be a debugging option or not. If the option is a debugging option, it will not be included in the options when building without debugging options or when building for final release.

Hide

If this property is true, the given command-line option will not appear in the list of option when showing the usage of the (un)installer. The option will still be included in the build, but the user will not be able to see that it is there.

Value(s)

This property has different meanings depending on the type of option.

Boolean

If the option is a boolean option, Values can specify a list of two values that translate to Yes and No. The user will still have to pass Yes or No on the command-line, but if Values is not null, InstallJammer will set the virtual text to the first value if the user specified Yes or the second value if they specified No.

Choice

If the option is a choice, Values must contain a list of possible values that are legal for the given option. The list can be anything you want, but the values themselves will be case-insensitive when the user passes them.

Prefix

If the option is a prefix, values contains a list of possible options the user can pass. If values is empty, the user can pass any option they want on the command-line with the prefix.

Switch

If the option is a switch, Values can contain a string that will be set in the virtual text if the user passes the option on the command line. If Values is null, the virtual text will be set to 1.

Description

This is a short description telling the user what this option is meant to do. When the user passes an incorrect option or specifies --help on the command-line, usage information will be displayed that contains all of the non-hidden options and their descriptions. The usage information for each option will be different for each option depending on its type, but the description can briefly tell the user what the option is for.

-0-

Virtual Definitions

Virtual Text

The Virtual Text panel allows you to see what virtual text and virtual directory definitions are defined in InstallJammer as well as add your own virtual text to your project. Adding your own virtual text lets you customize your installer by using common strings throughout instead of using constant values.

To learn more about virtual text, please read [What is Virtual Text?](#)

Adding New Virtual Text

You can add your own virtual text by clicking the "Add New Virtual Text" button located on the toolbar in the panel. A new line will be added to the bottom of the virtual text table with values that are left blank for you to fill in. The first field is the index value for the virtual text field you are adding. The second field is the text that will be used for your new virtual text within the installer.

Now that you've added your new virtual text, you can use it throughout your install in place of constants. Here's an example:

Let's say you add a virtual text definition called "MyVirtualText" with a value of "This is my virtual text." Anywhere you use `<%MyVirtualText%>` within InstallJammer, it will be substituted with the text "This is my virtual text." And, since virtual text can be embedded within other virtual text, you could set your virtual text to something like, "Today's date is `<%Date%>`" and InstallJammer will do the substitution of `<%Date%>` inside your text whenever you use it.

Deleting Virtual Text

You can only delete virtual text that you have added to your project. Virtual text that is defined by InstallJammer cannot be deleted. To delete a virtual text definition that you have added, click on the row for the text and then click the "Delete Virtual Text" button on the toolbar in the panel.

You can select multiple rows to delete at once.

-0-

Run Disk Builder

Disk Builder

The disk builder is where the actual installation binaries are built. Once you have completed all the steps to creating your install, you must build it in order to use it.

Building an install

Pressing the build button will start the build process. Only the platforms selected in the Platforms to build section will be built. This allows you to quickly rebuild a single platform instead of always having to build all of the binaries. Platforms that are not checked, or platforms that are not active will not be built.

Once the build process starts, you can continue working on the install, as the build process takes place in the background. This is convenient if your build is a long one, and you wish to keep making changes to your install.

Quick Build

A quick build means InstallJammer will rebuild all of the files for your installer, but it won't repack any of your application files. This is a fast way to rebuild changes you've made to the installer when you haven't actually changed any of the files you are installing.

Build for final release

This option tells InstallJammer that you are building your installs for final release. With this option checked, InstallJammer will rebuild the full install, including your application files, even if you select a quick build and disable debugging options in the resulting installer. This produces an installer that is ready for distribution to customers.

-0-

Test the Installation

Test Installer

This panel allows you to test your newly-built install. By default, testing an install means just executing the binary.

Save temporary directory for debugging

This option tells InstallJammer to save the temporary directory created during installation for debugging purposes. When an InstallJammer install runs, a temporary directory is created to hold the unpacking program and other temporary files. Normally, the temporary directory is cleaned up upon exiting the install. If this option is checked, the temporary directory will be left for debugging.

Test in default mode

This tells the install to use the Default install mode.

Test in console mode (UNIX only)

This tells the install to test the console-based version of the installer. InstallJammer will launch the installer in a new console window. Console installs are not available on Windows.

Test in silent mode

This tells the install to use the Silent install mode.

Test install without installing files

This option tells your install to run through all of the panes without actually going through the steps to install the files of your install.

Test install with an open console

This option tells the install to open a debugging console window during installation. The console allows you to do anything you want while your install is running.

-0-

Test Uninstaller

This panel allows you to test your uninstaller after you've run the installer on your system. InstallJammer will attempt to locate the uninstaller for the last installation of your project and run the uninstaller if it finds it.

Test in console mode (UNIX only)

This tells the uninstall to test the console-based version of the uninstaller. InstallJammer will launch the uninstaller in a new console window. Console uninstalls are not available on Windows.

Test in silent mode

This tells the uninstall to use the Silent uninstall mode.

Test install without uninstalling files

This option tells your uninstall to run through all of the panes without actually going through the steps to uninstall the files of your install.

Test uninstall with an open console

This option tells the uninstall to open a debugging console window during uninstallation. The console allows you to do anything you want while your uninstall is running.

-0-

Objects

Object Types

InstallJammer uses objects to define various pieces of an install. Almost anything that is stored in your install is an object. They are usually identified as having an ID that is the unique name of the object. Below is a list of object types and a brief description of what they do.

Action

See [What are Actions?](#)

Action Group

See [What are Action Groups?](#)

Component

See [Components](#).

Condition

See [What are Conditions?](#)

File Group

See [Groups and Files](#).

Files and Directories

See [Files and Directories](#).

Pane

See [What are Panes?](#)

Setup Type

See [Setup Types](#).

-0-

Standard Properties

Some properties are standard to many objects throughout InstallJammer. This is a list of standard properties that can appear throughout your project and what they are. An object is defined as any piece of an install (file group, file, action, condition, etc...).

ID

A unique identifier for this object. This ID is generated when the object is created and does not change throughout the life of the object.

Component

This property tells you what component was originally used to create this object.

Active

This tells InstallJammer whether this object is active or not. An inactive object is not packaged when building an installer.

Alias

An alias is an alias by which to call an object ID. Any object in InstallJammer that has an ID can also have an alias. This makes it easier to remember objects by their alias instead of their object ID. For example, the Install Actions group that is created for a new project is aliased to be called Install Actions to make it easier to call from other actions.

v

Comment

Comments are sometimes provided by InstallJammer to tell you what an object is doing, but they are usually set by someone building the project. Comments are not used by InstallJammer for anything and can contain any text you want.

Conditions

This tells you if there are any conditions placed on the object that will be checked during installation. Selecting this property will let you bring up the Conditions Editor to see and modify any conditions on the object.

Data

This property is used to hold user-specific data. Just like comments, this data is not used by InstallJammer and is safe for you to store anything you want into it. This can be helpful for storing other bits of relevant data with an object in the system for use by other objects, actions or conditions.

Include

This property tells InstallJammer when and where to include this object when running an installer or uninstaller. The object will either be: always included, included only when testing or included only when not testing. This property is affected by the <%Testing%> virtual text. When testing, only objects which are always included or included when testing will be created. When not testing, only objects which are always included or included when not testing will be created.

-0-

Panes

What are Panes?

Panes are what the user sees as they move forward and backward through your installer. Your available panes are defined by the theme you chose when you created your project, and each theme has a different set of panes and a different look-and-feel for those panes.

By default, the install theme you choose will select a default set of panes that are the most common for a simple installer. These usually include a welcome, a pane to let the user select their installation directory, the pane to copy the files and do the install, and a finishing pane. For most applications, these are the simplest panes you will need to give your user the minimum flexibility during installation.

Select a pane in your install, and you will see the properties for that pane come up to the right of the pane tree. These properties define the actions and look-and-feel of your pane. The most common options will already be chosen for you, but you can modify your panes any way you like.

Standard Properties

See [Standard Properties](#).

Advanced Properties

The advanced properties of a pane are different for each pane in each theme. Most are self-explanatory, but if you need help, you can usually find a brief description of the property by hovering over the property's name for a second, and a help balloon will appear.

Below is a list of some common properties that appear throughout the themes.

Anchor

This property tells InstallJammer where to place a window when displaying it. The coordinates are defined as the points of the desktop (n, s, e, w, etc...). So, s will place the window at the bottom center of the screen.

Buttons

Defines the buttons that are available on the selected pane. Each theme defines the buttons that are available on each pane and sets up the default values for each pane based on default settings. Any button not included in the list given will be hidden when the pane is displayed. Disabling a button is usually the better way to go to keep your installer uniform. This can be achieved with the [Modify Widget](#) action.

Icon

This property tells InstallJammer what image to use as a small icon on the selected pane.

Image

This property tells InstallJammer what image to use as a larger image on the selected pane.

Text Properties

Text properties are properties that directly relate to the text displayed in a pane. They have the added option of not substituting virtual text if it's not required or if you don't want it to.

Besides every button on the wizard, text properties also include any widget on a pane whose text can be changed. Anytime you change a text property, you can preview the pane to see what the new text will look like during installation.

-0-

Actions

What are Actions?

Actions are what InstallJammer uses to perform tasks while an installer or uninstaller is running. An action can be anything that you might want to do. Any action can be added to any pane in an install, and each action is checked and run in order during installation.

Actions allow you a great amount of flexibility in your installer. While many actions are provided as a convenience for common tasks you might want to perform, you also have the option to script your actions to do anything you like.

Each action may also have any number of conditions (see: [What are Conditions?](#)) attached to it that are checked before executing. If the action does not meet the necessary conditions, it will be skipped, and the next action will be checked.

Actions may also be a part of an action group (see: [What are Action Groups?](#)) where the group itself has a set of conditions that determine whether the group will be executed.

A new install will come setup with many default actions. Without actions, the installer is nothing more than a bunch of panes to move back-and-forth through. The actions are what really does the work during the installation.

Standard Properties

See [Standard Action Properties](#).

Supported Platforms

Most actions and conditions in InstallJammer are supported by all available platforms, but some of them are platform-specific and only apply to those platforms. When dealing with actions or conditions that are platform-specific, you don't need to worry about adding conditions to check the platform. When the documentation lists that only certain platforms are supported for an action or condition, InstallJammer will automatically skip those actions on other platforms during execution. The action will still show up in debug logs as being executed though, so if you wanted to make that clearer, you could add your own Platform Condition.

Advanced Properties

Each action has a different set of advanced properties that further define the properties of the action. Please see the documentation for each type of action for more information.

-0-

What are Action Groups?

Action groups are a way to group individual actions together under a single set of conditions. Action groups are defined independent of any particular pane and can be called from anywhere in an install using the [Execute Action](#) action.

When executing an action group, the conditions of the Execute Action are first checked, then any conditions on the action group itself are checked, and finally, if the Evaluate Conditions property of the Execute Action is true, the individual conditions on each action in the action group are checked before executing.

If any conditions on the Action Group exist that are executed Before the Next Action, these conditions are checked after all of the actions in the group are executed, and if the conditions fail, the entire action group will be executed again.



InstallJammer creates five action groups for a new project: Setup Actions, Startup Actions, (Un)Install Actions, Finish Actions and Cancel Actions.

Setup Actions

This action group is executed near the very beginning of the install and uninstall startup. It happens just before the command-line arguments specified by the user are parsed. Since no wizard exists yet and some of the setup has not been done yet, this action group should only be used to initialize virtual text or to alter command-line options before they are parsed.

Startup Actions

This action group is executed automatically during the startup of the installer. This is done after some default checks and setup has been done but before the first pane of the wizard is displayed.

(Un)Install Actions

This action group is called from within the (un)install panes and is usually attached to the Copy Files pane. InstallJammer does not call this action group automatically, instead letting you call it wherever you want from within the installer.

Finish Actions

This action group is executed automatically when the install is finished. It is the last thing called before the installer exits.

Cancel Actions

This action group is executed automatically when the install is cancelled. It is the last thing called before the installer exits.

Standard Properties

See [Standard Properties](#).

-0-

Standard Action Properties

ID

A unique identifier for this object. This ID is generated when the object is created and does not change throughout the life of the object.

Component

This property tells you what action was originally used to create this object.

Active

This tells InstallJammer whether this object is active or not. An inactive object is not packaged when building an installer.

Alias

An alias is an alias by which to call an object ID. Any object in InstallJammer that has an ID can also have an alias. This makes it easier to remember objects by their alias instead of their object ID. For example, the Install Actions group that is created for a new project is aliased to be called Install Actions to make it easier to call from other actions.

Comment

Comments are sometimes provided by InstallJammer to tell you what an object is doing, but they are usually set by someone building the project. Comments are not used by InstallJammer for anything and can contain any text you want.

Conditions

This tells you if there are any conditions placed on the object that will be checked during installation. Selecting this property will let you bring up the Conditions Editor to see and modify any conditions on the object.

Data

This property is used to hold user-specific data. Just like comments, this data is not used by InstallJammer and is safe for you to store anything you want into it. This can be helpful for storing other bits of relevant data with an object in the system for use by other objects, actions or conditions.

Execute Action

This tells InstallJammer when to execute the given action. The descriptions of when to execute mean just what they say. The default is After Pane is Displayed, which means to execute the action after the given pane has been displayed to the user.

Ignore Errors

This tells InstallJammer to ignore any errors that might occur while executing this action. Errors usually occur because of bad data, but they can be as a result of bugs. If this option is set to Yes, InstallJammer will keep moving to the next action / pane when it encounters an error instead of stopping the install.

-0-

Console Actions

Console Ask Yes or No

This action is used in a console install to ask the user a question that requires a Yes or No answer. The user is required to provide a valid answer before the installation continues.

Supported Platforms
UNIX

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Default

This property specifies the default value if the user hits Enter when asked the question. A default value of None means that the user must enter an answer.

Prompt

The prompt to display the user when asking the question.

Virtual Text

The virtual text variable (without <% and %>) to store the user's answer in. As per the rules of boolean values in virtual text, the result will be 0 if the user chooses No or 1 if they choose yes.

-0-

Console Clear Screen

This action is used in a console install to clear the screen.

Supported Platforms
UNIX

Standard Properties
See [Standard Action Properties](#).

-0-

Console Get User Input

This action is used in a console install to get a string of input from the user. The string is not checked for validity in any way, so it is up to you to use an action to check the user's response.

Supported Platforms
UNIX

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Echo Input

If this property is false, the input will not be echoed as the user types. This is useful for asking for passwords and other information that should not be seen on the screen.

Prompt

The prompt to display the user when asking the question.

Require Response

If this property is true, the user cannot enter empty data for a response. The action will continue to prompt for the information until a response is entered.

Trim Result

If this property is true, the user's result will be trimmed of all white spaces before it is set to the resulting virtual text.

Virtual Text

The virtual text variable (without <% and %>) to store the user's answer in. As per the rules of boolean values in virtual text, the result will be 0 if the user chooses No or 1 if they choose yes.

-0-

Console Message

This action is used in a console install to display a message to the user.

Supported Platforms
UNIX

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Allow Quit

If this property is true, the user will be allowed to quit reading the message if it is a long message and pagination is turned on. If the message all fits on the screen without paging, or if pagination is turned off, this property means nothing.

Paginate Message

If this property is true, a long message will be displayed to the user a page at a time to allow them to read it before continuing on to the next page.

Wrap Text

If this property is true, the text of the message will be wrapped, by word, to the size of the user's screen when it is displayed.

-0-

Console Pause

This action is used in a console install to pause before the next action is executed. The user is asked to press the space bar to continue.

Supported Platforms
UNIX

Standard Properties
See [Standard Action Properties](#).

-0-

Execute Actions

Execute Action

This action is used to execute another action or action group. The Execute Action's conditions are checked before executing the action specified, and then the conditions of the target action can be checked based on properties.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Action

The ID or alias of the action to execute. This can be either another action in the installer or an action group.

Evaluate Conditions

If this is true, the conditions on the target action will also be checked before executing the action. If this is false, only the conditions of the Execute Action action will be checked, and any conditions on the target action or action group will be ignored.

-0-

Execute External Program

This action will execute an external program like a shell script or batch file and return the results in specified virtual text. It can also be used to show progressive output while the external program is running, so that the user can see what the program is doing while it runs.



You cannot execute in a console or as root if Show Progressive Output is true.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Clear Progressive Widget

If this property is true, the progressive text widget used during Show Progressive Output will be cleared before the external program is executed.

Console Title (UNIX only)

If this property is not null, it specifies the title to use on the console window if the Execute in Console property is true.

Execute as Root (UNIX only)

If this property is true, InstallJammer will attempt to execute the given program as root. If the user is already root, the action will execute as normal. If the user is not root, they will be prompted for their root password which will then execute the program as root.

Execute in Console

If this property is true, InstallJammer will execute the external program in a console window. This is useful for executing scripts that expect some kind of input from the user on the command-line.



InstallJammer will attempt to locate the best possible terminal based on desktop environment. On KDE, this will be Konsole. On Gnome, it will be gnome-terminal. If nothing else can be found, good old xterm will be used.

If Execute in Console is specified in conjunction with Execute as Root, and the user is not root, the console will first ask the user for their root password in the same console window before executing.

Include Stderr

If this property is true, the standard error channel of the executing program will be included in the output from the program. This will include the standard error in the result of the execution or in the progressive output if it is being used.

Program Command Line

This is the command-line of the program to execute. The command line is executed exactly as it is input with all arguments included. Virtual text will automatically be substituted.

It should be noted that arguments on the command-line are separated by spaces, and each argument is passed separately to the system as arguments to the command being executed. Each argument is parsed individually for virtual text rather than substituting the string as a whole so that each argument is properly appended to a list of arguments to execute.

Progressive Output Widget

This property specifies a textbox widget that will be updated with the output of a progressive command as it becomes available. The textbox will automatically scroll down to keep up with new input. A widget can be specified by its ID or Alias if it was created with the [Add Widget](#) action, or it can be the name of a widget on the current pane, or it can be specified as <PANE>.<WIDGET>.

Result Virtual Text

This is a virtual text variable to store the result of external program. This will be any data that is output to standard out while the program is running. If progressive output is being shown, this variable will be the entire result that was output when the program finishes executing.

Show Progressive Output

If this value is true, the Progressive Output Widget will be updated with new data as it becomes available from the external program. If this value is false, all output from the external program is buffered and stored in Result Virtual Text.

Status Virtual Text

This is a virtual text variable that stores the exit code of the external program. An exit code of 0 usually means that the program succeeded without error. Any other exit code usually means that the program failed in some way.

Wait for Program

This tells InstallJammer to wait until the external program finishes before continuing on with the install. If this value is false, InstallJammer will execute the external program in the background and continue on.

Watch Progressive Output

If this property is true, and Show Progressive Output is true, InstallJammer will watch the output from the external program for cues that tell InstallJammer to set virtual text while it is running. The text InstallJammer looks for looks like a line surrounded by : with a single word as the first argument followed by any other arguments.

Example:

:Status Checking for file...:

This will set the <%Status%> virtual text variable to the value "Checking for file..." and then refresh the screen. This allows an external program to update the status or progress bars while it is running to keep the GUI alive so that the user knows what is going on while it is running. Any line that matches the output that InstallJammer is looking for will not be displayed in the output. It will be swallowed by InstallJammer before the user sees it.

Watch Regular Expression

This property specifies the regular expression to use when watching progressive output. By default, it looks for text in the manner described for Watch Progressive Output, but this property can be changed to look for any other kind of pattern. If the regular expression provides two submatch values, they will be used as the virtual text variable name and the value to set the virtual text to. If there is only one or no submatch, the Watch Virtual Text property will be used to store the value that matches. This property should only be changed if you really know what you're doing and know enough about regular expressions.

Watch Virtual Text

If the Watch Regular Expression does not include two submatches, this virtual text will be used as the virtual text to set with the result of the regular expression when watching progressive output.

Execute Script

This action is used to script unique actions in Tcl that allow you to do almost anything with your installer. Since Tcl scripts don't have any constraints within an installer, this action can be used to handle anything that is not already available in another action.

See What is Tcl/Tk? for more information.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Result Virtual Text

This is a virtual text variable to store the result of the script.

Tcl Script

This property holds the Tcl script to be executed. All virtual text will be substituted for the script.

-0-

File Actions

Adjust Line Feeds

This action is used to adjust the line feeds of the given files. This can convert text files to either UNIX (lf) or Windows (crlf) line feeds as specified.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Files

A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

Line Feed

The type of linefeeds for the specified files. The following choices are available:

- Auto - choose the linefeeds based on the target install platform.
- Unix - Use UNIX linefeeds (lf)
- Windows - Use Windows linefeeds (crlf)

Examples

The following values would convert all .txt and .sh files in <%InstallDir%> to UNIX linefeeds.

```
Files:      *.txt;*.sh  
Line Feed: Unix
```

-0-

Backup File

This action will take the specified files and back them up to a new file name. Given the new file extension, each file will be copied to a new file combined of its own file name plus the new extension.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Files

A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

File Extension

The new extension to append to each file being backed up. A %d in the string will be substituted for a number in the extension. The number will increment until an unused file name is found. Default is .bak%d

Overwrite Files

If this property is true, the action will overwrite a file that already exists when backing up instead of continuing to look for a new file name that does not exist. Default is No

Rename Files

If this property is true, the files specified will be renamed to the new file instead of just being copied. This is useful if you want to move the current file out of the way. Default is No

Starting Backup

The number to start at when backing up files. Default is 1

Examples

The following values would backup all .conf files in the <%InstallDir%>/conf directory.

Files: conf/*.conf

foo.conf would become foo.conf.bak1
bar.conf would become bar.conf.bak1

The following values would backup foo.cfg in <%InstallDir%> to foo.cfg.orig, but if the file already exists, nothing will happen and InstallJammer will throw an error. We will just tell InstallJammer to ignore the error.

Ignore Errors: Yes
Files: foo.cfg
File Extension: .orig

foo.cfg would become foo.cfg.orig

-0-

Change File Ownership

This action will change the ownership or group of a given list of files on the target system. The installing user must have permission to change the ownership to the given owner/group, or this action will fail.

Supported Platforms
All UNIX

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Files

A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

Owner

The new owner. This can be either a username or UID on the target system.

Group

The new group. This can be either a group name or GID on the target system.

Examples

The following values would change the ownership of every file in the <%InstallDir%> to owner root, group wheel.

```
Files: *  
Group: wheel  
Owner: root
```

-0-

Change File Permissions

This action will change the UNIX file permissions of the files given.

Supported Platforms

All UNIX

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Files

A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

Permissions

The UNIX permissions mask (in octal format).

Examples

The following values would make all of the .txt files in <%InstallDir%> read-only.

```
Files:          *.txt
Permissions: 0444
```

-0-

Copy File

This action will copy a single file or directory on the target system to another name.

Supported Platforms

All Platforms

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Source

The path to the file or directory to be copied. If the path is relative, it will be made relative to <%InstallDir%>.

Destination

The new path to copy the file or directory to. If the path is relative, it will be made relative to <%InstallDir%>.

Examples

The following values would copy the file foo.cfg to bar.cfg in the <%InstallDir%>.

```
Source:      foo.cfg
Destination: bar.cfg
```

-0-

Create File Link

This action creates a link in the file system to another file. The link can be a symbolic or a hard link.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Link Name

The file name of the link to create on the target system. This file will point to another file in the file system. If the path is relative, it will be made relative to <%InstallDir%>.

Link Target

The target file of the link to create. The new file link will point to this file. If the path is relative, it will be made relative to <%InstallDir%>.

Link Type

The type of link to create: hard or symbolic.

Examples

The following values create a symbolic link from our program executable into /usr/local/bin.

```
Link Name:    /usr/local/bin/myprogram
Link Target:  <%InstallDir%>/myprogram
Link Type:    symbolic
```

-0-

Create Folder

This action is used to create a new folder or directory on the target system.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Folder Name

The name of the folder or directory to create.

-0-

Delete File

This action will delete all of the files specified from the target system.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Files

A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

Examples

The following values would delete a /tmp/mytemp directory and all of the .bak1 files in the <%InstallDir%>.

Files: /tmp/mytemp;*.bak1

-O-

Read File Into Virtual Text

This action will read a single file on the target system into a virtual text variable. This is useful to obtain properties or information from a file and use it within InstallJammer.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Encoding

This property specifies the encoding to use when reading the file into virtual text. The default is to use whatever the system encoding is. utf-8 would be a common choice here. If you are unfamiliar with character encodings, it's probably best to leave this property alone.

File

The path to the file to read in. If the path is relative, it will be made relative to <%InstallDir%>.

Trim String

If this property is true, the data read from the file will be trimmed of whitespace on either side before the virtual text is set.

Virtual Text

The name of the virtual text variable to read the file into (without <% and %>).

Examples

The following values would read the file /etc/hosts.allow into the virtual text <%AllowedHosts%>

```
File:          /etc/hosts.allow
Virtual Text:  AllowedHosts
```

-0-

Rename File

This action will rename a single file or directory on the target system to another name. This is the same as "moving" a file.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Source

The path to the file or directory to be renamed. If the path is relative, it will be made relative to <%InstallDir%>.

Destination

The new path to rename the file or directory to. If the path is relative, it will be made relative to <%InstallDir%>.

Examples

The following values would rename the file foo.cfg to bar.cfg in the <%InstallDir%>.

```
Source:      foo.cfg
Destination: bar.cfg
```

-0-

Replace Text in File

This action will replace text strings in a file with other strings as based on a string map. This is useful for writing out configuration files with virtual text values from InstallJammer, etc...

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Encoding
Specifies the language encoding to use when replacing text in the file.

Files
A list of files or patterns separated by a ;. Any pattern or file that is a relative path is automatically made relative to <%InstallDir%>.

Line Feed
The type of linefeeds for the specified files. The following choices are available:

- Auto - choose the linefeeds based on the target install platform.
- Unix - Use UNIX linefeeds (lf)
- Windows - Use Windows linefeeds (crlf)

String Map
A string map is a list of strings in pairs separated by whitespaces. Each string in a string map has a corresponding string to map it to. Strings with spaces should be placed in quotes to show that they are to be treated as a single string. See the example below for a sample string map.

Examples

The following values would convert all instances of @@INSTALL_DIR@@ to our installation directory and all instances of the string "replace this string" with "our new string" in a file called config.cfg. It will also convert linefeeds to UNIX (lf) linefeeds.

```
Files:      config.cfg
Line Feed:  Unix
String Map: "@@INSTALL_DIR@" <%InstallDir%>
            "replace this string" "our new string"
```

-0-

Write Text to File

This action will write text out to a text file. It can either overwrite a file or append text to an existing file. Any file that does not exist will automatically be created.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

If this property is true, any file touched by this action will be added to the list of files to be removed during uninstallation.

Append Newline

If this property is true, a newline will be output at the end of the text to be written to the file.

Encoding

Specifies the language encoding to use when writing text to the file.

File Open Action

This tells the action how to open the file for writing. The following choices are available:

- Append to file - Append the text to an existing file or create a new one if it doesn't exist.
- Overwrite existing file - Write the text to a new file, overwriting any file that already exists.

Files

A list of files separated by a ;. Any file that is a relative path is automatically made relative to <%InstallDir%>.

Line Feed

The type of linefeeds for the specified files. The following choices are available:

- Auto - choose the linefeeds based on the target install platform.
- Unix - Use UNIX linefeeds (lf)
- Windows - Use Windows linefeeds (crlf)

Examples

The following values will write some of our install configuration out to a config.cfg file.

```
Files:      config.cfg
Line Feed:  Unix
```

```
Text to Write: InstallDir: <%InstallDir%>
                Installer:  <%Installer%>
                Username:    <%Username%>
```

-O-

Unzip File

This action will take a given .zip file that is installed on the target system and unzip it to a destination directory.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

If this property is true, any directory or file created while unzipping the file will be added the uninstall log to be removed later during uninstallation.

Destination

The new path to unpack the zip file to. If the path is relative, it will be made relative to <%InstallDir%>.

Progress Virtual Text

If not null, this specifies a virtual text variable to update with the percentage complete as the files are unzipped.

Status Virtual Text

If not null, this specifies a virtual text variable to update with the status of the files as they are unzipped. The virtual text will be set to the value of Status Text and then updated.

Zip File

The path to the zip file or directory to be unpacked. If the path is relative, it will be made relative to <%InstallDir%>.

Text Properties

Status Text

A text string to use when showing status during the unzip.

Examples

The following values would unpack the file foo.zip into the directory <%InstallDir%>/bar.

```
Source:      foo.zip
Destination: foo
```

-0-

General Actions

Continue Install

This action continues file installation if it was previously paused using the [Pause Install](#) action.

Standard Properties

See [Standard Action Properties](#).

-0-

Exit

This action will cause the installer to exit.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Exit Code

If this property is not empty, it specifies an exit code to exit the (un)installer with. By default, an (un)installer will exit with a 0 if it finishes successfully or 1 if the install is canceled.

Exit Type

This property tells InstallJammer what kind of exit this is. Whether it's a cancel or finish. This mostly affects what action group InstallJammer will execute as it is executing.

-0-

Fetch URL

This action will fetch a given URL from the web and store the result into a file on the local system. This is useful for downloading pieces of the installer that you don't want to package up but want to download when the user opts to.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Progress Virtual Text

The name of the virtual text variable to fill in with progress as the file is downloaded. This variable will hold the percent complete as the file is downloaded. You can use this on a pane with a progress bar to track the progress of the download.

Proxy Host

The hostname of a proxy server to use when making the request.

Proxy Port

The port on the proxy host to use when making the request.

Save To

Specifies that you want to save the fetched file to a file or a virtual text variable.

Target

If saving to a file, this is the name of the file or directory to save to. If this is a directory, the file name will be derived from the URL itself. If saving to a file, this property holds the virtual text variable (without <% and %>) to save the file contents to.

Timeout

The time, in microseconds, to wait before giving up when trying to connect to download the file.

URL

The URL to fetch.

-0-

Generate UID

This action will generate a GUID or UUID and store the result in a virtual text variable.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

UID Type

Whether to generate a GUID or a UUID.



A GUID is just a UUID with curly braces around it.

Virtual Text

The name of a virtual text variable (without <% and %>) to store the new UID into.

-0-

Launch File

This action will launch the given file on the target system using a default method for the file type of the given file. InstallJammer uses a default method on the target platform to launch the file based on its file type.



Files on UNIX are launched using xdg-open in the xdg-utils package that is part of the portland project by freedesktop.org.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Arguments

A list of arguments to pass on the command line when launching this file.

File Name

The name of the file on the target system to launch.

Wait for Program

Whether InstallJammer should wait for the program to finish and exit before continuing with the rest of the install.

Working Directory

If this property is not null, InstallJammer will change directory to this directory before launching the file.

-0-

Launch Web Browser

This action will launch a web browser to the specified URL on the target system. InstallJammer will attempt to use the default web browser for the target operating system. If it fails to find a default web browser, InstallJammer will attempt to find a suitable browser.

On Windows, the URL is launched using whatever the default browser is for the current user. On UNIX, if the desktop environment is KDE or Gnome, the default method for launching a URL will be used. If that method fails, or if the desktop is some other window manager, this action will attempt to find a suitable web browser based on environment and known browsers.



URLs on UNIX are launched using xdg-open from the xdg-utils package that is part of the portland project by freedesktop.org.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

URL

The URL to point the web browser to when it launches.

-0-

Log Debug Message

This action will log a message to the debug log and to the console when an installer is run in debug mode.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Message
The message to output to the debug log.

-0-

Message Box

This action will pop up a message box that requires the user's attention. It can be used simply to give the user notice of something, or it can be used to ask the user a question and get an answer. Once posted, InstallJammer will not continue with the install until the user has clicked a button to close the message box.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Default Button

The button to focus on by default. This is the default action if the user simply hits Enter when the message box pops up.

Icon

A choice of icon to display in the message box giving the user an indication what type of message this is.

Type

The type of message box to display. There are several common types to choose from. Each type will display with the given list of buttons.

Virtual Text Field

The virtual text variable to hold the result of the message box. This variable will contain the name of the button the user clicked on. The name of a button is the text displayed on the button in all lowercase. So, if the user clicks Yes, the virtual text is set to yes.

-0-

Message Panel

A message panel is a borderless, buttonless panel that pops up to display some message to the user. This is usually done when the installer is doing something that may take a moment, and you want the user to see what is happening. Once a message panel is posted, unless it is given a timeout, it must be destroyed using a [Destroy Widget](#) action.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Anchor

The position on the screen to display the message panel. Default is the center of the screen.

Grab Focus

This tells InstallJammer to grab the focus from the user so that they can't interact with anything else in the installer while the message panel is on the screen.

Height

The height of the message panel in pixels.

Image

An icon to display on the left side of the message panel.

Image Pad X

How many pixels to pad the image on the left and right from the rest of the panel. A single number means to pad the image on both sides by that amount. Two numbers means to pad the image on the left the first number of pixels and pad the right the second number of pixels.

Image Pad Y

How many pixels to pad the image on the top and bottom from the rest of the panel. A single number means to pad the image on both sides by that amount. Two numbers means to pad the image on the top the first number of pixels and pad the bottom the second number of pixels.

Label Anchor

The anchor for the text in the label.

Label Pad X

How many pixels to pad the label on the left and right from the rest of the panel. A single number means to pad the label on both sides by that amount. Two numbers means to pad the label on the left the first number of pixels and pad the right the second number of pixels.

Label Pad Y

How many pixels to pad the label on the top and bottom from the rest of the panel. A single number means to pad the label on both sides by that amount. Two numbers means to pad the label on the top the first number of pixels and pad the bottom the second number of pixels.

Timeout

The number of seconds the panel should stay on the screen. If this value is 0, the panel will stay on the screen until it is destroyed with a [Destroy Widget](#) action.

Width

The width of the message panel in pixels.

-0-

Modify Object

This action is used to modify the properties of any object in an installer. See [Object Types](#) for a description of the available object types.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Object State
If specified, this says that the given object should be Active or Inactive.

Object ID
The ID or alias of the object to be modified.

-0-

Pause Install

This action will pause the file installation that is currently in progress. Once paused, the install must either be stopped using a [Stop Install](#) action or continued using a [Continue Install](#) action.

Standard Properties

See [Standard Action Properties](#).

-0-

Set Object Property

This action will set the value of any property on a given object.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Object ID

The ID or alias of the object to set the property on.

Property

This is the property to set on the given object.

Substitute Value

If this property is true, the Value will be substituted fully for virtual text before being set as the new property value.

Value

The value to set the property to on the object.

-0-

Set Virtual Text

This action will set the value of a virtual text variable.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Auto Update

If this property is true, the virtual text being set will be added to a list of variables that automatically update the screen whenever they are changed. By default, InstallJammer only updates the screen during actions or when moving between panes. This will make this virtual text cause an update anytime its value changes.

Language

This property tells InstallJammer what language to set the virtual text in. If the value is None, the virtual text is set in a non-localized variable that works for all languages. If the virtual text already exists in a language though, that value will override the non-localized value.

Value

This is the new value to set the virtual text variable to.

Virtual Text

The name of the virtual text variable to set.

-o-

Stop Install

This action stops file installation if it was previously paused using the [Pause Install](#) action.

Standard Properties

See [Standard Action Properties](#).

-0-

Text Window

This action pops up a simple text window to display a long bit of text to the user. It is useful for posting text files like READMEs and for displaying progressive output from an [Execute External Program](#) action.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Anchor

The position on the screen to display the message panel. Default is the center of the screen.

Grab Focus

This tells InstallJammer to grab the focus from the user so that they can't interact with anything else in the installer while the message panel is on the screen.

Height

The height of the text window in pixels.

Image

An icon to display on the left side of the message panel.

Text File

The name of a file on the target system to read and display in the text window.

Text Wrap

Tells the textbox how to wrap text.

Wait on Window

If this value is true, InstallJammer will wait for the user to close the window before continuing on with the installation. If it is false, the text window will pop up, and the install will continue on.

Width

The width of the text window in pixels.

-O-

Wait

This action will cause InstallJammer to wait a specified number of milliseconds before continuing with the install. This works just like sleep on most systems.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Wait Time

The amount of time (in milliseconds) to wait. 1000 == 1 second.

-o-

Install Actions

Add Install Info

This action is used to add your own information to the InstallJammer registry for the installed application. This information can be retrieved later through a [Check for Previous Install](#) action or just by reading the install registry.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Key

The name of the property to store to the registry. This should be a short key containing alpha numeric characters.

Value

The value to store in the registry under Key. This value will automatically be substituted for virtual text before it is stored to the registry.

Example

The following values would record the install mode the user used during installation.

```
Key:    Mode
Value:  <%InstallMode%>
```

Then, when using Check for Previous Install, you would have a virtual text <%PreviousInstallMode%> that equals whatever mode the user installed with (Console, Silent, Standard, etc...).

-0-

Add Response File Info

This action is used to add virtual text variables to the list of variables to be saved to a response file when writing. If the user has passed `--save-response-file` on the command line, signifying that they want to save the responses from the installer to a file on exit, only virtual text which has been designated to be saved will be saved. Use this action to add virtual text to the list of what to save.



The default virtual text variables saved with every response are:

- CreateDesktopShortcut
- CreateQuickLaunchShortcut
- InstallDir
- InstallMode
- InstallType
- LaunchApplication
- ProgramFolderName
- SelectedComponents
- ViewReadme

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Virtual Text

This is a list of virtual text variables separated by `;`. Each virtual text can also be specified with a type that will tell the installer how it should be saved to a file and read back in. The current types supported are described below

boolean	the virtual text will be saved as "Yes" or "No" depending on its current value
list	the virtual text is considered a valid Tcl list and will be saved as a comma-separated list

When specifying your list of virtual text, you can specify the type by adding it after the virtual text name.

Example

Virtual Text: Foo; Bar list; MyBool boolean

This would add the three virtual text variables: Foo, Bar and MyBool to the list of variables to be stored in the response file. Foo would be stored as-is with no interpretation. Bar would be stored as a list, which means that `<%Bar%>` contains a valid Tcl list, and it will be stored as a comma-separated list in the response file. MyBool is a boolean, so the installer will take whatever value is in `<%MyBool%>` and determine if it is true or false and convert its value in the response file to Yes or No. Any boolean value is perfectly valid within a response file, but Yes and No are used to make it easier on the user to read the file.

-O-

Add to Uninstall

This action will add a directory, file or registry entry to the install log so that InstallJammer will attempt to uninstall it during uninstallation. This allows you to add things to the uninstall that might be generated by your application but you don't want to bother the user with deleting themselves.



If this action is used during uninstallation before the uninstall actually begins, the directory, file or registry key will be added to the list of files about to be uninstalled.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Component Name

The name of the component to uninstall. This is either the name of a file or directory, or the name of a Windows registry key to uninstall.

Component Type

This tells InstallJammer what type of component is being added to the uninstall.

Forcefully Remove Directory

If this value is true, InstallJammer will delete the directory regardless of whether it has anything in it or not. By default, InstallJammer will not delete a directory that is not empty.

Registry Value

If this value is non-empty, it tells InstallJammer to remove a value of the specified registry key instead of the registry key itself. If this value is left blank, InstallJammer will uninstall the entire registry entry.

-0-

Check for Previous Install

This action checks the target system for a previous install of your application and sets up virtual text variables as a result. When searching for a previous install, InstallJammer uses the Application ID of your project to find them.

The following virtual text variables are set as a result of this action:



Version 1.0 of InstallJammer only defined the `<%PreviousInstallExists%>` and `<%PreviousInstallDir%>` variables. You should use a [Virtual Text Exists Condition](#) if you are unsure of whether the previous install virtual text you want exists.

If `<%PreviousInstallExists%>` is false, none of the rest of this virtual text is set.

`<%PreviousInstallApplicationID%>`

The Application ID of the previously installed application. This should match your own Application ID unless the previous install was an upgrade install. In that case, the Upgrade Install ID should match your application ID.

`<%PreviousInstallDate%>`

The install date of the last install in clock seconds. This can be formatted using the `<%Date%>` virtual text.

`<%PreviousInstallCount%>`

The number of previous installations found on the target system. Only the last install, by date, will be used for this information.

`<%PreviousInstallDir%>`

The installation directory of the last install.

`<%PreviousInstallDirExists%>`

A convenience value that tells you whether or not the previous install dir still exists on the target system.

`<%PreviousInstallExecutable%>`

The full path of the previous install executable when it was run.

`<%PreviousInstallExists%>`

This value will be true if a previous install was found, or false if one was not found.

`<%PreviousInstallID%>`

The Install ID of the last install. This is the ID that is generated everytime an installer is run, so this ID will change with each subsequent install.

`<%PreviousInstallIDs%>`

This virtual text contains a list of all previous install IDs that have been installed for this application.

`<%PreviousInstallRealUser%>`

The real username of the person who installed last. This is the real username even if the user was running as root.

`<%PreviousInstallSource%>`

The directory where the previous installer was run from.

`<%PreviousInstallUninstaller%>`

The full path to the uninstaller of the last install.

<%PreviousInstallUpgradeID% >

The Upgrade Application ID of the last install.

<%PreviousInstallUser% >

The username of the person who installed last.

<%PreviousInstallVersion% >

The install version (1.0.1.0, etc...) of the last install.

<%PreviousInstallVersionString% >

The install version string (1.0a1, etc...) of the last install.



Other variables can also be set if a developer has added information to the install registry through the [Add Install Info](#) action. Any variable added through that action will also appear in virtual text as a result of this action. For example:

If the Key Foo was added with the Value Bar, you would get the variable
<%PreviousInstallFoo%> that would equal Bar.

Standard Properties

See [Standard Action Properties](#).

-0-

Get Previous Install Info

This action gets a specific set of install information based on a given Install ID. Since a single application can have multiple installations on the same system, this action can be used to get specific information from a specific install. You must first find the Install ID though.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Application ID

The Application ID to read previous install information from. This defaults to the current installer's <%ApplicationID%>, but this action can also be used to read install information from other installations.

Install ID

The install ID of the install whose information you want to get.

Virtual Text Prefix

A string to prefix to the beginning of each piece of information from the install registry. This prefix will be used for each variable from the registry to add information to virtual text variables depending on what information is available.

The following virtual text variables are set as a result of this action (assuming that your Virtual Text Prefix is PreviousInstall):

<%PreviousInstallApplicationID% >

The Application ID of the previously installed application. This should match your own Application ID unless the previous install was an upgrade install. In that case, the Upgrade Install ID should match your application ID.

<%PreviousInstallDate% >

The install date of the last install in clock seconds. This can be formatted using the <%Date%> virtual text.

<%PreviousInstallCount% >

The number of previous installations found on the target system. Only the last install, by date, will be used for this information.

<%PreviousInstallDir% >

The installation directory of the last install.

<%PreviousInstallDirExists% >

A convenience value that tells you whether or not the previous install dir still exists on the target system.

<%PreviousInstallExecutable% >

The full path of the previous install executable when it was run.

<%PreviousInstallExists% >

This value will be true if a previous install was found, or false if one was not found.

<%PreviousInstallID% >

The Install ID of the last install. This is the ID that is generated everytime an installer is run, so this ID will change with each subsequent install.

<%PreviousInstallRealUser% >

The real username of the person who installed last. This is the real username even if the user was running as root.

<%PreviousInstallSource% >

The directory where the previous installer was run from.

<%PreviousInstallUninstaller% >

The full path to the uninstaller of the last install.

<%PreviousInstallUpgradeID% >

The Upgrade Application ID of the last install.

<%PreviousInstallUser% >

The username of the person who installed last.

<%PreviousInstallVersion% >

The install version (1.0.1.0, etc...) of the last install.

<%PreviousInstallVersionString% >

The install version string (1.0a1, etc...) of the last install.



Other variables can also be set if a developer has added information to the install registry through the [Add Install Info](#) action. Any variable added through that action will also appear in virtual text as a result of this action. For example:

If the Key Foo was added with the Value Bar, you would get the variable <%PreviousInstallFoo%> that would equal Bar.

-0-

Install Log File

This action installs a copy of the InstallJammer install log into a target location. By default, InstallJammer installs this log in a registry directory that it uses for subsequent installs and uninstallation, but this action will let you create a copy of it somewhere else on the system as well.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Log File Location

This is the directory and filename where you want to install this log file.

-O-

Install Selected Files

This is the main action of the installation process. This action starts up the actual file installation process and handles installing all of the files the user has selected to install during the process. This action is created in the Install Actions group by default, which is executed from the Copy Files pane of a new install project, but you can move it anywhere you like or remove it all together if your installer is not actually installing any files.

This action will build up a list of groups and files to be installed based on the current Setup Type. The Components of the Setup Type are first checked, and any Component that is inactive will be skipped. Then, the File Groups beneath the Component are checked, and any File Group that is inactive will be skipped. Finally, any file contained in the File Groups that is inactive will be skipped.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Show Console Progress

If this property is true, a progress bar will be shown during console installs.

Update File Text

If this property is true, the text properties on the progress pane will be updated every time a new file begins installation. By default, InstallJammer shows what group is being installed and not each and every file. Updating the window with each file can be expensive and can take longer than just updating when a new group is being installed.

Update File Percent

This should only be set to true if you have two progress bars on the same pane and one is tracking the progress of each individual file using `<%FilePercentComplete%>`. Again, as noted above, it can be very expensive and take longer to install if you are updating on every single file change.

-0-

Install Uninstaller

This action creates an uninstaller for the current installation on the target system. This action must come after the [Install Selected Files](#) action, as that is the action that builds the list of files, directories and registry entries to be uninstalled.



Be careful installing more than one uninstaller on the target system. InstallJammer will register its package information with the last uninstaller installed, and that is what it will use during uninstallation.

It is usually best to only install one instance of an uninstaller.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Message

This is the message that is used as the status while the uninstall is being built.

Uninstall Directory

This specifies what directory you want to create the uninstaller in. By default, it is created in the main installation directory.

Uninstall Filename

The file name to use for the uninstaller on the target system. This defaults to uninstall with the appropriate extension.

-0-

Install Wish Binary

This action creates a Tcl/Tk Wish binary on the target system. It is not actually a standard Wish binary but a single-file executable wish that is built from a base installkit.



The installkit is a stand-alone WISH shell that contains the following:

Tcl 8.4.14
Tk 8.4.14
Tile 0.7.8
Tkpng 0.7
Itcl 3.3
Thread 2.6.4 (Windows only)

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Binary Name

This is the name of the wish binary you wish to create.

Windows Icon

On Windows, this specifies a .ico file to use as the icon for this binary. If no Windows Icon is specified, the wish binary will have the same icon as the original installer.

-O-

Install Wrapped Script

This action takes a Tcl/Tk script and wraps it into a self-contained binary executable on the target system.



The .tcl script is wrapped with the same installkit that built the installer. It uses the internal installkit::wrap command to wrap the script into its own executable.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Binary Name

This is the name of the target binary on the target system.

Company

On Windows, this specifies the Company name to store in the Windows Resource information of the file.

File Description

On Windows, this specifies the File Description to store in the Windows Resource information of the file.

File Version

On Windows, this specifies the File Version to store in the Windows Resource information of the file.

Include TWAPI

If this property is true, and TWAPI is included in the current installer, the TWAPI extension will be included in the wrapped script when it is built. The specified script will be changed to include some code at the top of the script to load the TWAPI extension on startup, so it is not necessary to package require the extension.

Installkit

The path to the installkit to be used to wrap the script. If this is left blank, the current running installer will be used as the base installkit.

Product Name

On Windows, this specifies the Product Name to store in the Windows Resource information of the file.

Product Version

On Windows, this specifies the Product Version to store in the Windows Resource information of the file.

Tcl Startup Script

This is the .tcl script to wrap into the binary.

Windows Icon

On Windows, this specifies a .ico file to use as the icon for this binary. If no Windows Icon is specified, the wish binary will have the same icon as the original installer.

-o-

Unpack Stored File

This action will unpack a file stored in the installer to a specified location on the target system. Since the installer will already unpack all of the selected files during installation, this action should only be used to extract a file to a special location or before the actual installation is performed.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

If this property is true, the unpacked file will be added to the list of files for the uninstaller to remove.



Files unpacked to the <%Temp%> directory are never added to the uninstall.

File ID

The ID of the file to unpack. This can be the valid ID of any file stored in the installer or an alias to a file.

Target Directory

The target directory to unpack this file into.

Target Filename

The name of the file to create. If this property is left blank, the name of the file will be the same name that it was stored with.

-0-

Java Actions

Get Java Property

This action will retrieve a specified property value from a standard Java properties file.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Property

The name of the property to get from the property file.

Property File

The path to the property file to open.

Virtual Text

The name of the virtual text variable to fill with the property if it is found in the property file. If the property is not found in the property file, the value of the virtual text will be blank.

-0-

Locate Java Runtime

This action attempts to locate the most suitable Java Runtime Environment (JRE) available on the target system. It does this by looking at some default paths and environment variables on the system as well as allowing you to specify locations to check.

If the action successfully finds a suitable JRE, the following virtual text variables are set as a result:

<%JavaAvailableVersions% >

A list of the available versions found on the target system before finding the correct one. Once an acceptable JRE is found, InstallJammer stops looking, so this list will only include up to and including the JRE found that matches our needs.

<%JavaExecutable% >

The full path to the java executable (usually <%JavaHome%>/bin/java).

<%JavaFound% >

True if a Java runtime was found or false if none was found.

<%JavaHome% >

The Java home directory.

<%JavaVersion% >

The version of the Java runtime found. (1.4.2_11, etc...)

<%JavaVersionMajor% >

The major version of the Java runtime found. (1.4, 1.5, etc...)

<%JavaVersionMinor% >

The minor version of the Java runtime found. (1.4.2, 1.6.0, etc...)

<%JavacExecutable% >

The full path to the javac executable if it is found. If a javac executable is not found, this virtual text will not be set.

<%JavawExecutable% >

The full path to the javaw executable if it is found. If not found, this variable will point to the main java executable instead.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Maximum Version

This property tells InstallJammer the maximum version of the JRE you will accept.

Minimum Version

This property tells InstallJammer the minimum version of the JRE you will accept.

Require JDK

If this property is true, the action will require that a full JDK (Java Development Kit) be installed and not just a JRE (Java Runtime Environment).



This will make the installer search for the existence of a javac (the Java compiler that is

lacking in a JRE installation) binary to determine if the installation is a JDK.

Search Path

This property tells InstallJammer how to search for the JRE. The value is a list of paths to check for a suitable java installation (separated by ;).

If Default Search Path is included as a path, InstallJammer will use defaults like environment variables and the user's PATH on the target system to try and find the JRE.



The Default Search Path searches in the following order:

- JAVA_HOME environment variable
- JAVAHOME environment variable
- JDK_HOME environment variable
- JRE_HOME environment variable
- JAVA_ROOT environment variable
- JAVA_BINDIR environment variable
- PATH environment variable
- Windows registry (Windows only)
- Program Files/Java/jre* (Windows only)
- /usr/java/jdk (UNIX only)
- /usr/lib/java (UNIX only)
- /usr/lib/jvm (UNIX only)
- /usr/lib/jvm/jre (UNIX only)
- /usr/lib/java-1.4.0 (UNIX only)
- /usr/lib/java-1.4.1 (UNIX only)
- /usr/lib/java-1.4.2 (UNIX only)
- /usr/lib/java-1.5.0 (UNIX only)

If Prompt User is included as a path, InstallJammer will check all of the other paths in order, and upon reaching Prompt User, it will prompt the user asking them to specify the location of the runtime environment. This is usually best left as the last path so that all of the other directories will be checked before finally prompting the user when no JRE has been found.

-0-

Shortcut Actions

Install Desktop Shortcut

This action installs a shortcut on the desktop of the target system depending on the system and desktop environments found. On UNIX systems, this action looks for the existence of both KDE and Gnome and will install the shortcut on either or both desktops if available.



If the user is root when they run the install, the shortcut will be added for all users on the system. If they are not root, the shortcuts will only be added for the user doing the installation.



UNIX shortcuts are installed using the xdg-icon-desktop utility from the xdg-utils package that is part of the portland project by freedesktop.org.

Supported Platforms

All

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Additional Arguments

This property contains any of a number of lines to be added to the UNIX desktop file after the standard properties that are setup by InstallJammer. Each line will be substituted for virtual text and then appended to the desktop file without any further modification.

Command Line Arguments

This property specifies arguments to be passed to the target filename on the Windows platform.

Icon Path

This property specifies the file to use for the icon for this shortcut. On Windows, this can be a .ico, .exe or any other file containing icon information. On UNIX platforms, this usually points to a .png file to display.

Icon Path Index

The index into the file that Windows uses when looking for an icon.

Install for All Users

On Windows, if this option is true, the desktop shortcut will be installed for all users and not just the current user.

Shortcut Name

The name to display for the shortcut.

Shortcut Type

On UNIX, this specifies what type of shortcut this is. It can be a shortcut to a URL Link, or a shortcut to an Application.

Target File Name

The name of the file this shortcut points to. On a UNIX shortcut that has a type of Link, this

is the URL to point to.

Working Directory

On Windows, this specifies the working directory that the target of this shortcut will start in.

Vendor ID

The Vendor ID is used to group all of the shortcuts and program folders together on UNIX platforms. By default, everything is grouped by the Application ID of the current installer, but setting this to a common value would allow for all of the installers in your company to share program folders, for example.

-O-

Install Program Folder Shortcut

This action installs a shortcut into the Program Folder for your application on the target system depending on the system and desktop environments found. On UNIX systems, this action looks for the existence of both KDE and Gnome and will install the shortcut under either or both menus if available.



If the user is root when they run the install, the shortcut will be added for all users on the system. If they are not root, the shortcuts will only be added for the user doing the installation.



UNIX shortcuts are installed using the xdg-icon-menu utility from the xdg-utils package that is part of the portland project by freedesktop.org.

Supported Platforms

All

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Additional Arguments

This property contains any of a number of lines to be added to the UNIX desktop file after the standard properties that are setup by InstallJammer. Each line will be substituted for virtual text and then appended to the desktop file without any further modification.

Command Line Arguments

This property specifies arguments to be passed to the target filename on the Windows platform.

Icon Path

This property specifies the file to use for the icon for this shortcut. On Windows, this can be a .ico, .exe or any other file containing icon information. On UNIX platforms, this usually points to a .png file to display.

Icon Path Index

The index into the file that Windows uses when looking for an icon.

Install for All Users

On Windows, if this option is true, the desktop shortcut will be installed for all users and not just the current user.

Shortcut Name

The name to display for the shortcut.

Shortcut Type

On UNIX, this specifies what type of shortcut this is. It can be a shortcut to a URL Link, or a shortcut to an Application.

Target File Name

The name of the file this shortcut points to. On a UNIX shortcut that has a type of Link, this is the URL to point to.

Working Directory

On Windows, this specifies the working directory that the target of this shortcut will start in.

Vendor ID

The Vendor ID is used to group all of the shortcuts and program folders together on UNIX platforms. By default, everything is grouped by the Application ID of the current installer, but setting this to a common value would allow for all of the installers in your company to share program folders, for example.

-O-

Install UNIX Program Folder

This action creates a desktop shortcut file on a UNIX system.



If the user is root when they run the install, the folder will be added for all users on the system. If they are not root, the folder will only be added for the user doing the installation.



UNIX program folders are installed using the xdg-utils package that is part of the portland project by freedesktop.org.

Supported Platforms

FreeBSD, Linux

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Additional Arguments

This property contains any of a number of lines to be added to the .directory file after the standard properties that are setup by InstallJammer. Each line will be substituted for virtual text and then appended to the desktop file without any further modification.

Folder Directory

This is the directory to install this shortcut in if Folder Location is set to Directory.

Folder Location

Specifies whether you want to create the .directory file in some directory of your choosing or under the standard program folder.

Folder Name

The name of the folder to create on the target system.

Icon Path

This property specifies the file to use for the icon for this shortcut. This usually points to a .png file to display.

Vendor ID

The Vendor ID is used to group all of the shortcuts and program folders together. By default, everything is grouped by the Application ID of the current installer, but setting this to a common value would allow for all of the installers in your company to share program folders, for example.

-0-

Install UNIX Shortcut

This action creates a desktop shortcut file on a UNIX system.



If the user is root when they run the install, the shortcut will be added for all users on the system. If they are not root, the shortcuts will only be added for the user doing the installation.



UNIX shortcuts are installed using the xdg-utils package that is part of the portland project by freedesktop.org.

Supported Platforms

FreeBSD, Linux

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Additional Arguments

This property contains any of a number of lines to be added to the UNIX desktop file after the standard properties that are setup by InstallJammer. Each line will be substituted for virtual text and then appended to the desktop file without any further modification.

Command Line Arguments

This property contains arguments that should be appended to the target filename to form the full command line for the shortcut. This differs from the Additional Arguments in that the additional arguments are lines that are directly appended to the .desktop file when it is created on the target system. The command line arguments are concatenated with the target filename to form the command line to execute.

File Name

The name of the desktop file on the target system.



The File Name property cannot be empty.

Icon Path

This property specifies the file to use for the icon for this shortcut. This usually points to a .png file to display.

Program Folder Name

If the shortcut being installed is going into the Program Folder, this is the name of the program folder to put it into. By default, it uses the program folder for the rest of the installation.

Shortcut Directory

This is the directory to install this shortcut in.

Shortcut Location

Where to install the shortcut. This can be either a directory, the Desktop or the Program Folder. If the shortcut is to be installed in a directory, a .desktop file is simply created in the shortcut directory with the given values. If the shortcut is to be installed on the Desktop, the Shortcut Directory is ignored, and the file is installed on the Desktop. If the shortcut is to be

installed in the Program Folder, it will be installed in a program folder under the specified Program Folder Name.

Shortcut Name

The name to display for the shortcut.

Shortcut Type

This specifies what type of shortcut this is. It can be a shortcut to a URL Link, or a shortcut to an Application.

Target File Name

The name of the file this shortcut points to. If this shortcut has a type of Link, this is the URL to point to.

Vendor ID

The Vendor ID is used to group all of the shortcuts and program folders together. By default, everything is grouped by the Application ID of the current installer, but setting this to a common value would allow for all of the installers in your company to share program folders, for example.

-O-

Install Windows Shortcut

This action installs a shortcut on the desktop of the target system depending on the system and desktop environments found. On UNIX systems, this action looks for the existence of both KDE and Gnome and will install the shortcut on either or both desktops if available.



If the Target File Name looks like a URL (http://, ftp://, etc...) the shortcut will be a URL shortcut. If the Target File Name does not look like a URL, it is assumed to be a file on the target system, and the file must exist. If the file does not exist, no shortcut will be created.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Command Line Arguments

This property specifies arguments to be passed to the target filename. These arguments are appended to the Target File Name when creating the shortcut so that when the shortcut is executed, the Target File Name is called with the given arguments.

Description

This property sets the Comments property of the Windows shortcut, which is also the tooltip that is displayed if the mouse is hovered over the shortcut.

Icon Path

This property specifies the file to use for the icon for this shortcut. This can be a .ico, .exe or any other file containing icon information.

Icon Path Index

The index into the file that Windows uses when looking for an icon.

Shortcut Directory

This is the directory to install this shortcut in.

Shortcut Name

The name to display for the shortcut.

Target File Name

The name of the file this shortcut points to.

Working Directory

This specifies the working directory that the target of this shortcut will start in.

-0-

System Actions

Add Directory to Path

This action is used to add a directory to a path variable of the target system.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

If this property is true, the added paths will automatically be removed from the system during uninstallation.

Directory

A list of directories separated by a ; to be added to the path variable.

Level

This specifies whether the directory should be added to the path variable for every user on the system or just the current user.

Location

This specifies whether the given directory should be prepended to the beginning of the path variable or appended to the end.

Normalize Paths

If this property is true, each path added to the variable will be normalized appropriately for the current platform before it is added. This makes sure that any path that is relative or has incorrect directory separators is made to be legal for the current platform. Default is Yes.

Separator

This property specifies what separator to use between the paths for the given variable. Defaults to <%PathSeparator%>.

Variable

This property specifies the variable to add your path to. By default, this is PATH, but it could easily be something like CLASSPATH to add to the Java class path on the target system.

-o-

Add Environment Variable

This action is used to add a new environment variable to the target system.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

If this property is true, the added paths will automatically be removed from the system during uninstallation.

Level

This specifies whether the variable should be added for every user on the system or just the current user.

Restore on Uninstall (Windows only)

This property tells InstallJammer to remember the value of the environment variable if it already exists and then restore it during uninstall. This is only valid for Windows because this is always the case on UNIX systems because of the way variables are added.

Value

The value to set the new environment variable to.

Variable

This property specifies the name of the environment variable to add to the target system.

-0-

Delete Environment Variable

This action is used to delete an environment variable from the target system.

Supported Platforms

Windows

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Level

This specifies whether the variable should be deleted for every user on the system or just the current user.

Variable

This property specifies the environment variable to be deleted from the target system.

-0-

Reboot or Shutdown System

This action will reboot or shutdown the target system. On some UNIX systems, this action cannot be performed unless the user is root.

Supported Platforms

All

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Execute as Root

If this property is true, InstallJammer will prompt the user for their root password if they are not already root.

Type

Whether the system should be rebooted or shutdown.

-0-

Register Package

This action will attempt to register a package with the given package manager on the target system. This action will almost always require root to be performed. If the user is not already root, the action will attempt to execute the action as root which will prompt the user to enter their root password to install the package information.



Currently-supported package systems are: DPKG and RPM. The installer will attempt to build a dummy package on the target system using standard tools and then install it. Because of the limitations of the build tools, this action does not register the files of your installer with the package database.

It does, however, register the uninstaller with the package system so that if the user attempts to remove the package through their system, it will call the uninstaller to do the actual file removal.



InstallJammer will automatically add this package to be removed from the package database during uninstallation if the package is not being removed from the package manager itself. This requires root access, and InstallJammer will prompt the user to execute the removal as root if they are not root when they run the uninstall.

Supported Platforms

Linux

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

DPKG Changelog File

A file on the target system to use as the changelog file for the DPKG database. If no file is specified, InstallJammer will create a dummy file to satisfy DPKG. If the file does not have a .gz extension, InstallJammer will gzip the file as-per DPKG requirements.

DPKG Control Script

This is the script that will be used as the control script when DPKG builds our dummy package. The default values and information will build a standard package based on information in your installer.



Unless you are familiar with building DPKG packages, it is not recommended that you change any of the values in the Control Script. This can lead to your package not building or not installing correctly.

DPKG Copyright File

A file on the target system to use as the copyright file for the DPKG database. If no file is specified, InstallJammer will create a dummy file to satisfy DPKG. If the file does not have a .gz extension, InstallJammer will gzip the file as-per DPKG requirements.

DPKG Debian Changelog File

A file on the target system to use as the Debian changelog file for the DPKG database. If no file is specified, InstallJammer will create a dummy file to satisfy DPKG. If the file does not have a .gz extension, InstallJammer will gzip the file as-per DPKG requirements.

DPKG Pre Install Script

A script to execute before installation in the DPKG database. Since InstallJammer is doing most of the work here, you should probably leave this blank unless you know what you are doing.

DPKG Pre Uninstall Script

A script to execute before uninstallation in the DPKG database. The default script is a standard script expected for all DPKG packages, and you should probably not change it unless you know what you are doing.

DPKG Post Install Script

A script to execute after installation in the DPKG database. The default script is a standard script expected for all DPKG packages, and you should probably not change it unless you know what you are doing.

DPKG Post Uninstall Script

A script to execute after uninstallation in the DPKG database. InstallJammer automatically populates this script with a script to call our uninstaller. Since the DPKG database doesn't really know anything about the files in our installer or how to remove them, we want DPKG to remove its package and then call our uninstaller.

Since DPKG attempts to remove a previously-installed package before proceeding with a new install, this script checks to see if our removal is being run from another installer and exits if we are.

Package Name

The name of the package to use when storing this package to the database. The rest of the information is stored in the different scripts that build our dummy packages.

Package Databases

Specifies which package database should be registered with on the target system. Defaults to all, which means both RPM and DPKG if they exist on the system.

RPM Spec Script

The contents of this script are stored to a SPEC file to build our dummy RPM from. The default values and information will build a standard package based on information in your installer.



Unless you are familiar with building RPM packages, it is not recommended that you change any of the values in the Spec Script. This can lead to your package not building or not installing correctly.

-O-

Remove Directory from Path

This action is used to remove a directory from a path variable on the target system.

Supported Platforms

Windows

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Directory

The directory to be removed from the path variable.

Level

This specifies whether the directory should be removed from the path variable for every user on the system or just the current user.

Separator

This property specifies what separator to use between the paths for the given variable.

Defaults to <%PathSeparator%>.

Variable

This property specifies the variable to add your path to. By default, this is PATH, but it could easily be something like CLASSPATH to add to the Java class path on the target system.

-0-

Uninstall Actions

Uninstall Leftover Files

This action is just like [Uninstall Selected Files](#) except that it will forcefully try to remove any pieces of the uninstall that could not be removed on the first pass. This means deleting directories eventhough they are not empty or forcibly trying to remove a file that could not be deleted the first time.

If any errors still occur while executing this action, they are appended to the <%Errors%> virtual text, and the <%ErrorsOccurred%> virtual text will be set to true.

Standard Properties

See [Standard Action Properties](#).

-0-

Uninstall Selected Files

This is the main action of the uninstall process. This action will attempt to uninstall everything that was installed on the target system during installation. This includes all files, directories and registry entries.

If any piece cannot be removed for some reason, an error will be appended to the <%Errors%> virtual text, and the <%ErrorsOccurred%> virtual text will be set to true.

Standard Properties

See [Standard Action Properties](#).

-0-

Windows Actions

Disable Wow64 Redirection

On 64 bit Windows Vista (and beyond?) systems, installers and setup programs have certain system directories automatically redirected to a 32 bit directory hierarchy. This action disables that redirection so that libraries can be properly installed in the 64 bit directories.

Disabling the redirection will continue for the rest of the installation or until a [Revert Wow64 Redirection](#) action is executed, but the redirection only applies to the current process, so it will not continue once your installer exits.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

-O-

Revert Wow64 Redirection

On 64 bit Windows Vista (and beyond?) systems, installers and setup programs have certain system directories automatically redirected to a 32 bit directory hierarchy. This action reverts any previous redirection from the [Disable Wow64 Redirection](#) action.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

-0-

Windows Registry Actions

Add Windows File Command

This action will add a new command to a file type on the target Windows system. This means that a new command will appear in the right-click menu for this file type.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Command

This is the actual command to execute when someone clicks on the menu entry for this command.

Command Name

This is the name of the command to be added. Usually something like Open, Edit or Print.

File Type

This is the file type to add this command to on the target system.

Menu Name

If specified, this sets the title to display on the menu for this command. If this property is empty, the Command Name will be used as the menu title.

-0-

Add Windows File Extension

This action adds a new file extension to the Windows registry and attaches it to a given file type.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

File Extension

This is the file extension to add. It should include the ., so it would be something like .tcl or .mpi.

File Type

This is the file type to add this command to on the target system.

-0-

Add Windows File Type

This action adds a new File Type to the Windows registry of the target system.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

File Type

The name of the file type to be added.

Icon Path

This property specifies the file to use for the icon for all files of this file type. This can be a .ico, .exe or any other file containing icon information.

Icon Path Index

The index into the file that Windows uses when looking for an icon.

Show File Extensions

Tells Windows whether to display the file extensions for all files of this file type. If not specified, files of this type will behave according to the system or folder defaults.

Title

The title of this file type. This title is displayed in the Type column of Windows Explorer for all files matching this file type.

-o-

Add Windows Registry Key

This action adds a new registry key and/or value to the Windows registry on the target system.



By default, this action will add the new registry key to the uninstall, and it will be removed along with the files and directories when the application is uninstalled. This can be prevented by setting Add to Uninstall to No.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Add to Uninstall

This property tells InstallJammer whether or not to add this registry key to the list of things to be uninstalled. If it is set to No, the key will not be uninstalled with the rest of the application.

Data Type

This property specifies the type of data that will be stored in the registry key.

Key

The name of the key to create in the registry.

Name

The name of a value to add to this registry key. If this property is empty, the value will be set as the default value for this registry key.

Value

The actual value of the data stored in the value being added to this registry key.

Restore on Uninstall

If this property is true, InstallJammer will check if the registry key already exists on the system before installing. If the registry key / value already exists, the value will be remembered and then restored during uninstallation.

Root Key

The root key in the registry to add this key to.

-0-

Add Windows Uninstall Entry

This action will add a new entry to the Windows Add or Remove Programs registry. This can be accessed from the Control Panel and is the usual way for Windows users to remove installed applications from their system.

InstallJammer adds one of these actions to a new install by default so that Windows users will have a way to remove your application in a way that is already familiar to them.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Comments

This property stores comments for this entry in the Windows registry.

Contact

This property specifies contact information displayed to the user.

Display Name

This property tells Windows what to display in the Add or Remove Programs window for this application.

Display Icon

This property specifies the file to use for the icon for this entry. This can be a .ico, .exe or any other file containing icon information.

Display Version

This property tells Windows what version to display for this program.

Help Link

A URL to the homepage of your application.

Help Telephone

Your contact telephone number for support.

Install Date

The date of installation in the format of %Y%m%d.

Install Location

The installation directory.

Install Source

The directory the installer was originally run from.

Publisher

The name of the company or person publishing this application.

Quiet Uninstall String

The command to execute to quietly uninstall your application.

Readme

The location of the README file for your application.

Registry Key Name

The key to use in the uninstall registry. By using <%ApplicationID%>, it ensures that subsequent installs of your application will always use the same uninstall registry entry instead of adding a new one for every version of your software.

Uninstall String

The command to execute to uninstall your application.

URL Info About

A URL that points to information about your application.

URL Update Info

A URL that points to information about updating your application.

-0-

Import Windows Registry File

This action will import a Windows registry (.reg) file into the Windows registry on the target system.



This action simply calls regedit with the registry file to do the import. The registry file must be in a standard format.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Prompt User

If this property is true, Windows will prompt the user to ask if they want to import this file before beginning. The user can opt not to import the registry file.

Registry File

The name of the file on the target system to be imported.

-0-

Register Windows Library

This action will register a DLL or other library with the Windows registry.



This action simply builds up the arguments and then calls the regsvr32.exe that is standard on Windows systems.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Call DLL Install

If this property is true, the DLLInstall() function will be called for the given library when it is installed.

Call DLL Register Server

If this property is true, the DLLRegisterServer() function will be called for the given library when it is registered. If this property is false, then Call DLL Install must be true.

Library File

The path to the library file to register.

-0-

Remove Windows Registry Key

This action will remove a registry key or value from the target system.



You do not need to remove during uninstall any registry keys and values added during installation. InstallJammer automatically registers any keys added by its own installers and adds them to the list to be uninstalled.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Key
The name of the key to remove from the registry.

Value
A value to be removed from the key. If this is not null, the key itself will not be removed, only this value will be removed from the key.

Root Key
The root key in the registry. If Key already contains a root key, this property is ignored.

-0-

Unregister Windows Library

This action will unregister a DLL or other library with the Windows registry.



This action simply builds up the arguments and then calls the regsvr32.exe that is standard on Windows systems.

Supported Platforms
Windows

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Call DLL Uninstall

If this property is true, the DLLUninstall() function will be called for the given library when it is installed.

Call DLL Unregister Server

If this property is true, the DLLUnregisterServer() function will be called for the given library when it is registered. If this property is false, then Call DLL Uninstall must be true.

Library File

The path to the library file to unregister.

-O-

Windows Service Actions

Continue Windows Service

This action will continue a Windows service that is in a paused state.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

Wait

If not null, this specifies a number of milliseconds to wait for the action to complete. InstallJammer will wait until the action completes or for the specified number of milliseconds, whichever comes first.

-0-

Create Windows Service

This action will continue a Windows service that is in a paused state.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Account

Specifies an account name to be the user account under which the service should run. The account name takes the form domain\username. If the account is a local account, the account name may be specified as .\username or username. If this option is not specified, the service will run under the LocalSystem account. When Service Type is Kernel Driver or File System Driver, the account name should be the name of the driver object that the system uses to load the driver.

Command

This is the system command to be executed to start the service.

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Dependencies

Specifies the services and service load order groups that are required by this service and that must start before this service. This property contains a list of service names and load order groups. A load order group name must be prefixed with a + character to indicate that it is a group name and not a service.

Description

Specifies the long description for the service when viewed within the services console.

Display Name

Specifies the user visible name for the service. This is the name that is shown in the net start command and the SCM control panel applet.

Error Control

This option specifies how errors during service start are to be handled.

Critical	An error message will be logged to the Windows event log if possible. If the last known good configuration is active, the system startup is aborted. Otherwise, the system is restarted with the last known good configuration.
Ignore	An error message will be logged to the Windows event log. The system startup will continue.
Normal	An error message will be logged to the Windows event log and a popup displayed to the user. The system startup will continue. This is the default.
Severe	An error message will be logged to the Windows event log. The system startup will continue only if the last known good configuration is active. Otherwise, the system is restarted with the last known good configuration.

Interactive

This option specifies whether the service should be allowed to interact with the desktop. This is only valid when Service Type is specified to be Own Process or Shared Process. The Account option must not be used with this option since interactive services must run under the LocalSystem account.

Load Order Group

Specifies a group name to be the service load order group to which the service belongs. If the option is not specified, the service is assumed to not belong to any load order group.

Password

Specifies the password corresponding to the user account specified in the Account option. This option is ignored if the Account option is not specified.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

Service Type

This property specifies the type of service to create.

Own Process	Service that runs in its own process. This corresponds to the SERVICE_WIN32_OWN_PROCESS in the Windows SDK documentation. This is the default if the option is not specified.
Shared Process	Service that runs inside a process shared with other services. This corresponds to the SERVICE_WIN32_SHARED_PROCESS in the Windows SDK documentation.
File System Driver	File system driver. This corresponds to the SERVICE_FILE_SYSTEM_DRIVER in the Windows SDK documentation.
Kernel Driver	Kernel driver. This corresponds to the SERVICE_KERNEL_DRIVER in the Windows SDK documentation.

Start Type

This property specifies how the service is to be started.

Auto Start	The service should be automatically started by the service control manager during system startup. This is the default if the option is not specified.
Boot Start	The service should be automatically started by the system loader during system boot. This is only valid when Service Type is specified to be Kernel Driver or File System Driver.
Demand Start	The service should be started by the system upon receiving an explicit program request.
Disabled	The service is disabled and cannot be started even on program request.
System Start	The service is a driver that is started by the operating system IoInitSystem function during system initialization. This is only valid when Service Type is specified to be Kernel Driver or File System Driver.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

-0-

Delete Windows Service

This action will delete a Windows service from the target system.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

-0-

Pause Windows Service

This action will pause a Windows service.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

Wait

If not null, this specifies a number of milliseconds to wait for the action to complete. InstallJammer will wait until the action completes or for the specified number of milliseconds, whichever comes first.

-0-

Start Windows Service

This action will start a Windows service.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

Wait

If not null, this specifies a number of milliseconds to wait for the action to complete. InstallJammer will wait until the action completes or for the specified number of milliseconds, whichever comes first.

-0-

Stop Windows Service

This action will stop a Windows service.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

Wait

If not null, this specifies a number of milliseconds to wait for the action to complete. InstallJammer will wait until the action completes or for the specified number of milliseconds, whichever comes first.

-0-

Wizard Actions

Add Pane to Order

This action will take the given pane and add it to the wizard's back order at the given index. This is useful for controlling how the wizard behaves in tricky installs.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Index

This property specifies where in the current back order of the wizard this pane should be added. This can be a number, the word end, or another pane ID. If the index is a number, it means that the given pane should be inserted just before that index in the order. If the index is the word end, it means to append the pane to the end of the order. If the index is another pane, it means that the given pane should be inserted just before that pane in the current order.

Pane

The ID or alias of a pane to be added to the back order of the wizard.

-0-

Add Widget

This action will add a new widget to an install pane. Widgets are any type of control that is drawn on a pane, which includes: buttons, checkbuttons, labels, radiobuttons and more.



Setting the Text on any widget type other than a label will cause a label to be drawn in combination with the specified widget. This is the easiest way to add a label with a widget rather than having to add a separate label widget.

Supported Platforms

All

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Action

This property specifies an action that should be executed if the user clicks a button or modifies the value of a checkbutton or radiobutton. The action also applies to entry, browse entry and password entry widgets when the Validate Entry On property is set.

Background

The background color of the new widget. The most common are system, which is the main system color for the background of windows, and white.

Browse Type

If the widget is a browse entry, this property specifies what you are browsing for. A directory will use a choose directory dialog. An open file will use a file dialog that is expecting a file that already exists. A save file will use a file dialog that will tell the user the file will be overwritten if it already exists.

Checked

When adding a radiobutton or checkbutton, this property tells InstallJammer that the button should be checked on by default when displayed. If this property is left blank, the status of the button will depend on the virtual text property attached to it. In the case of a checkbutton, if the virtual text property is true, the checkbutton will be checked. In the case of a radiobutton, the radiobutton whose value matches the current value of the virtual text will be checked.

Editable

If the widget type is a combobox, this property specifies whether the value in the entry of the combobox should be editable or not. If the value is not editable, the user may only choose from the given list of values. If the combobox is editable, the user can either choose from the given values or type in their own value.

File Types

If the widget type is a browse entry for choosing files, this property tells the file chooser popup what file types to allow. The File Types property must contain a list of pairs containing a description of the file type and a list of extensions to include with the file type. Example: "Source Files" ".c .cpp .h .hpp" "Text Files" ".txt" "All Files" ""

Foreground

The foreground color of the new widget. The most common are system, which is the main system color for the foreground text of windows, and black.

Height

The height of the widget. Usually, this option should be left blank, and InstallJammer will make the widget as tall as it needs to be.

Label Side

On any widget that is not a label, setting the Text property will cause a label to be drawn with the corresponding widget. This property tells InstallJammer where you want the label to appear in relation to the widget. The choices are on top or to the left.

Label Width

This property specifies the width of the text label when it is present next to another widget. Using this property with your label on the left side, you can make all of your labels line up evenly by giving them all the same width.

Off Value

This is the value the virtual text will be set to when a checkbox is off. This is only valid for checkboxes.

On Value

This is the value the virtual text will be set to when a checkbox is on. This is only valid for checkboxes.

Text

If the widget type is a label, the text property specifies what should be displayed in the label. If the widget is not a label, setting the text property to any string will cause a label to be drawn with the given widget.

Type

The type of widget to add. The list currently includes: browse entry, button, checkbox, combobox, entry, label, label frame, password entry, radiobutton or text.

Validate Entry On

This property tells InstallJammer when to validate the contents of an entry widget when the type of widget added is an entry, and the action property is not empty. The following values are acceptable as either a single value or as a list of values separated by ; . Create, Focus, FocusIn, FocusOut and Key. Create means that the value will be validated when the widget is created. Focus, FocusIn and FocusOut mean the entry is validated when the focus of the entry changes. Key means the entry will be validated anytime a key is pressed inside the entry.

Value

For a radiobutton, this is the value stored in the virtual text when this radiobutton is selected. Please read the section titled Boolean Values in Virtual Text in [What is Virtual Text?](#) for information on using boolean-type values as a value for a radiobutton.



Setting the value on an entry or browse entry will set the default value of the virtual text to the given value, which will make the value appear in the entry box.

Values

If the widget type is a combobox, this property specifies a list of values to display in the dropdown menu of the combobox. This must be a valid list with each item quoted. Example: "Item a" "Item b" "Item c" "Item d"

Virtual Text

The virtual text variable to store the result in. Radiobuttons are grouped together by all sharing the same virtual text variable. The selected radiobutton will set the value of the virtual text.

Width

The width of the widget. If this option is left blank, InstallJammer will make the widget as wide as it needs to be.

X

The x coordinate on the pane to place this widget.

Y

The y coordinate on the pane to place this widget.

-O-

Append Text to Widget

This action will append text to a text widget in the current pane displayed.

Supported Platforms

All Platforms

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Auto Scroll

If this property is true, the text widget will automatically scroll to the bottom after the text is appended.

Clear Widget

If this property is true, the text widget will be cleared of any text before adding the new text.

Widget

The name of the text widget to append to.

-0-

Create Install Panes

This action will create all of the install panes for the installer. This is usually done at the front of the install to create all of the panes at once so that they display more quickly as the user moves through them. If the panes are not created up front, they will be created as each one is displayed for the first time.

If this action is placed on a pane with a progress bar, it will increment the progress bar as each pane is created.

Standard Properties

See [Standard Action Properties](#).

-0-

Destroy Widget

This action will permanently destroy a widget. A widget can either be a single widget on a pane created with the [Add Widget](#) action, or it can specify a message panel or another pane.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Widget ID

A list of Widget IDs or Aliases separated by ;. A widget ID can also be specified as <PANE>.<WIDGET> where <PANE> is the ID or Alias of a specific pane and <WIDGET> is the ID or Alias of a widget on that pane.

-O-

Focus On Widget

This action will draw the keyboard focus to a widget.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Widget ID

The ID or alias of the widget to be destroyed.

-0-

Modify Widget

This action will modify the text or state of a given widget or list of widgets on the current pane.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

State

The state the modified widget should be. Normal means that the widget is active, and the user can interact with it. Disabled means that the widget is inactive, and the user cannot manipulate it. Disabled widgets are usually shown as "greyed out" to reflect that they are disabled. Hidden means that the widget should be removed from the pane entirely. A hidden widget can be restored by modifying its state back to normal. Readonly is a state that can be applied to entry and text widgets and means that the widget will appear as normal, but the user will not be able to enter any data.

Text

The text the modified widget should display. This will modify all the listed widgets and change their text to the given value. If left blank, the text will not be modified.

Widget

This property specifies a single widget on the current pane, or a list of widgets separated by a semicolon (;).

-o-

Move Forward

This action causes InstallJammer to move forward to the next pane in the install. If there are no more panes to be displayed, it will cause InstallJammer to exit with a Finish condition.

Standard Properties

See [Standard Action Properties](#).

-0-

Move to Pane

This action causes InstallJammer to jump forward or backward through the installer to the specified pane. When this action moves to the given pane, that pane will automatically be added to the backwards order of the wizard so that the user will see it if they go back through the wizard.



If the pane being moved to is the current pane, the actions of the pane will be executed from the start, effectively reloading the current pane.

Standard Properties

See [Standard Action Properties](#).

Advanced Properties

Pane

The ID or alias of the pane to jump to.

-O-

Populate Components

When placed on a Choose Components pane, this action will populate the tree with a list of components for the currently-selected setup type. Anytime the setup type changes (like the user moves back), the tree will be cleared and re-populated.

Supported Platforms

All

Standard Properties

See [Standard Action Properties](#).

-0-

Populate Setup Types

When placed on a Setup Type pane, this action will populate the setup type list with the setup types included in the current installer.

Supported Platforms
All

Standard Properties
See [Standard Action Properties](#).

-0-

Remove Pane from Order

This action will take the given pane and remove it from the wizard's back order. This means that when the user moves back with the Back button, this pane will no longer be shown. This is useful for controlling how the wizard behaves in tricky installs.

Supported Platforms
All Platforms

Standard Properties
See [Standard Action Properties](#).

Advanced Properties

Pane
The ID or alias of a pane to be added to the back order of the wizard.

-0-

Conditions

What are Conditions?

Conditions are a way to tell InstallJammer how and when it should do certain things within an installer. Most objects in InstallJammer can have conditions attached to them that will control whether or not an action is taken on that object.

When conditions are checked, they are checked with an explicit AND, which means that the first false condition that is found will stop, and the rest will not be checked. The [Script Condition](#) can be used in cases where more complicated conditions are needed.

-0-

Standard Condition Properties

ID

A unique identifier for this object. This ID is generated when the object is created and does not change throughout the life of the object.

Component

This property tells you what condition was originally used to create this object.

Active

This tells InstallJammer whether this object is active or not. An inactive object is not packaged when building an installer.

Alias

An alias is an alias by which to call an object ID. Any object in InstallJammer that has an ID can also have an alias. This makes it easier to remember objects by their alias instead of their object ID. For example, the Install Actions group that is created for a new project is aliased to be called Install Actions to make it easier to call from other actions.

Check Condition

This property tells InstallJammer when to check this condition. This property is only valid for panes and actions and does not appear for conditions applied to other object types. The possible choices are different depending on if you are currently modifying a pane or an action.



When the Check Condition property on an Action is set to Before Next Action is Executed, it will cause the current action to be executed again if the condition fails. This is mostly useful during Console installations where you want the action to execute again to display its message and prompt the user for input. The user cannot continue to the next action until the condition is satisfied.

Comment

Comments are sometimes provided by InstallJammer to tell you what an object is doing, but they are usually set by someone building the project. Comments are not used by InstallJammer for anything and can contain any text you want.

Data

This property is used to hold user-specific data. Just like comments, this data is not used by InstallJammer and is safe for you to store anything you want into it. This can be helpful for storing other bits of relevant data with an object in the system for use by other objects, actions or conditions.

Failure Focus

A widget to move the focus to if the condition fails. If Failure Message is not empty, the message will be displayed, and then the focus will be moved to the Failure Focus widget. If no message is displayed, the focus will simply move to the Failure Focus widget.

Failure Message

A message to display to the user if this condition is checked and fails. If this property is left blank, no message will be displayed to the user.

-o-

File Conditions

File Exists Condition

This condition checks to see if a file exists or does not exist on the target system.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Filename

The name of the file on the target system.

Operator

exists	True if the file does exist.
does not exists	True if the file does not exist.

-0-

File Name Test Condition

This condition checks a filename to see if it contains any invalid characters and determines if the filename is a legal name.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Filename

The name of the file on the target system.

Operator

name is valid	True if the filename is a legal name.
name is not valid	True if the filename is not a legal name.
path is valid	True if the each directory in the filename path is a legal name.
path is not valid	True if any directory in the filename path is not a legal name.

-0-

File Permission Condition

This condition checks the permissions of a file on the target system.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Filename

The name of the file on the target system.

Permission

can create	True if the file or directory can be created on the target system.
cannot create	True if the file or directory cannot be created on the target system.
is readable	True if the file exists and is readable.
is not readable	True if the file does not exist or is not readable.
is writable	True if the file exists and is writable.
is not writable	True if the file does not exist or is not writable.

-0-

General Conditions

Command Line Test Condition

This condition can check values passed by the user on the command line

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

exists

True if the command line option exists.

does not exist

True if the command line option does not exist.

was passed on the command line

True if the option was passed on the command line.

was not passed on the command line

True if the option was not passed on the command line.

Option

The name of the option to test. Note that any leading - or / characters will be removed before checking.

-0-

Execute Script Condition

This condition is a Tcl script that must return true if the condition passes or false if the condition does not pass.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Script

The Tcl script to evaluate. All virtual text is substituted when evaluating the script. Your script should always end with a return command that returns either true or false. Anything that is returned that looks to be true will be true. Anything that does not look to be true (including an empty string or other text) will be considered false and cause the condition to return false.

Example

The following script is an example that returns true if we are running a patch installer or false otherwise:

```
if {[string is true <%RunningInstaller%>] && [string is true  
<%UpgradeInstall%>]} {  
    return 1  
}  
return 0
```

Note that the script always returns a true or false answer. This is the safest way to ensure that your condition will always return something valid to the installer.

-0-

Object Test Condition

This condition checks to see if an object in the (un)install is currently active.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Object ID

The ID or alias of the object you are checking.

Operator

exists	True if the object exists.
does not exist	True if the object does not exist.
is active	True if the object is active.
is not active	True if the object is inactive.

-0-

Script Condition

This condition is a Tcl script that is substituted and then evaluated by Tcl's expression parser. This is not a condition that is used for full scripts. It is used for a combination of expressions that are combined together. To use an actual Tcl script as a condition, use the [Execute Script Condition](#).

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Script

A Tcl expression to evaluate. The entire expression is substituted for virtual text before being evaluated.

Example

The following example is a valid Tcl expression:

```
[string is true <%RunningInstaller%>] && [string is true <%UpgradeInstall%>]
```

Note that this is an expression, not a full Tcl script. An expression is multiple commands that are strung together using expression operators like && and ||. The following is not a valid expression:

```
set test1 [string is true <%RunningInstaller%>]
set test2 [string is true <%UpgradeInstall%>]
return [expr {$test1 && $test2}]
```

This is a full Tcl script to be evaluated, not an expression. This type of script can be used in the [Execute Script Condition](#), but not in a Script Condition.

-0-

Virtual Text Test Condition

This condition checks to see if a virtual text variable exists or not.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Language

The language of the virtual text you are checking. None means to check the virtual text without a language.

Operator

exists	True if the virtual text exists.
does not exist	True if the virtual text does not exist.
was set from the command line	True if the virtual text was set or modified as a result of an option passed on the command line.

Virtual Text

The virtual text (without <% and %>) to check.

-0-

Platform Conditions

Platform Condition

This condition checks to see if a file exists or does not exist on the target system.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

is	True if the platform is the current install platform.
is not	True if the platform is not the current install platform.
matches	True if the platform matches the given wildcard pattern. (Linux*) would match all Linux platforms.
does not match	True if the platform does not match the given wildcard pattern.

Platform

The name of a platform or pattern to match.

-O-

String Conditions

String Equal Condition

This condition checks to see if two given strings equal each other. Each string is substituted for virtual text before comparison.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

equals	True if the two strings equal each other.
does not equal	True if the two strings do not equal each other.

String 1

The first string to compare.

String 2

The second string to compare.

-0-

String Is Condition

This condition checks the type of a given string to see if it is or is not of the correct type.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

true	True if the sting is true (1, true, yes or on).
false	True if the string is false (0, false, no or off).
is empty	True if the string is empty, meaning it contains no real data.
is not empty	True if the string has any data in it.

String

The string to check. The value of the string will be substituted for virtual text before comparison.

-0-

String Match Condition

This condition checks to see if a given string matches a given pattern using standard glob-style matching. Each string is substituted for virtual text before comparison.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

matches True if the string matches the pattern.
does not match True if the string does not match the pattern.

Pattern

The pattern to match the string against.

The *pattern* argument may contain any of the following special characters:

?

Matches any single character.

*

Matches any sequence of zero or more characters.

[*chars*]

Matches any single character in *chars*. If *chars* contains a sequence of the form *a-b* then any character between *a* and *b* (inclusive) will match.

x

Matches the character *x*.

{*a,b,...*}

Matches any of the strings *a*, *b*, etc.

String

The string to match against the pattern.

-0-

System Conditions

Env Variable Test Condition

This condition checks to see if an environment variable exists or not.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

exists	True if the virtual text exists.
does not exist	True if the virtual text does not exist.

Variable

The name of the environment variable to check.

-0-

Package Test Condition

This condition checks to see if a package is installed in the target system's default package manager. This condition will determine what the default package manager is for the target system and then attempt to determine if the given package is installed by querying the package manager.



Currently-supported package systems are: DPKG and RPM.



RPM will use 'rpm -q \$package'

and

DPKG will use 'dpkg -s \$package'

and then check the exit code.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

is installed	True if the package is installed.
is not installed	True if the package is not installed.

Package

The name of the package to check.

-0-

Port Test Condition

This condition checks to see if a given port number on the target system is currently bound and in use. It does not make any checks to see what is running on that port, only that the port is bound and answering or not.



The action will attempt to open a socket to 127.0.0.1 (the local host) on the given port and see if anything answers the connection attempt. It then closes the port without ever sending or receiving any data.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

- can bind True if the port is not currently bound and can be bound.
- cannot bind True if the port is currently bound and cannot be bound.

Port

The port number to check.

-0-

User Input Conditions

Ask Yes or No

This condition posts a message box that asks the user a question and expects a yes or no response.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Default Button

The button to highlight as the default when the message box posts. This will be the button clicked if the user hits Return.

Icon

The standard icon to use for the message box.

Result Virtual Text

The virtual text variable to hold the result of the message box.

True Value

Specifies which value of the message box will return a true value for the condition. If the user clicks Yes, and Yes is the true value, this condition will pass, etc...

-0-

Windows Conditions

File Extension Test Condition

This condition checks to see if a given file extension exists in the Windows registry.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

File Extension

The extension to check.

Operator

exists	True if the extension exists.
does not exist	True if the extension does not exist.

-0-

File Type Test Condition

This condition checks to see if a given file type exists in the Windows registry.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

File Type

The file type to check.

Operator

exists	True if the file type exists.
does not exist	True if the file type does not exist.

-0-

Registry Test Condition

This condition checks to see if a given registry key or value exists in Windows registry.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Operator

exists	True if the key or value exists.
does not exist	True if the key or value does not exist.

Registry Key

The registry key to check.

Registry Value

The registry value to check. If this property is null, we only check to see if the key exists.

-0-

Windows Service Test Condition

This condition tests certain properties of a Windows service.

Standard Properties

See [Standard Condition Properties](#).

Advanced Properties

Advanced Properties

Database

Specifies the service control database to be operated on. By default, this is the active service control database on the target system.

Operator

exists	True if the service exists.
does not exist	True if the service does not exist.
is paused	True if the service is in a paused state.
is running	True if the service is currently running.
is stopped	True if the service is not currently running.

Service Name

The name of the service to perform the action on. This can be the internal or the display name of the service. If the display name is given, it will automatically be translated to the internal service name for the action.

System

Specifies the name of the system on which the command should be invoked. By default, this is the local system.

-0-

Virtual Text

What is Virtual Text?

Virtual text is used throughout InstallJammer as a way to define variables within an install and sometimes within the install builder itself. You will see virtual text throughout your use of InstallJammer as text that is between `<%` and `%>`.

Some virtual text is defined by you when you create your project and other virtual text is defined by InstallJammer when it builds your project. An example of virtual text that you define is the `<%AppName%>` virtual text which represents the Application Name you filled in when you created your project. An example of virtual text defined by InstallJammer would be `<%InstallMode%>` which, during installation, can tell you whether the user chose a Standard, Default or Silent install.

Virtual text is an easy way for InstallJammer to represent variable data within an install, and since it can be nested, virtual text can contain other virtual text that will all be represented when it is used within an install.

Using Virtual Text

Virtual text can be used almost anywhere in an install, and InstallJammer will automatically substitute the virtual text for its value when it is needed. To use virtual text, simply use one of the available [Virtual Text Definitions](#) between `<%` and `%>`.

For example, `<%AppName%>` when used in a project will be substituted for your Application Name during installation. `<%Platform%>` will be substituted for the name of the current platform you're installing on (Windows, Linux-x86, etc...).

Virtual text can be nested inside of another virtual text definition, and InstallJammer will automatically substitute all the values recursively. This means that if your Application Name contains "My Application `<%Version%>`" InstallJammer will substitute the `<%Version%>` recursively inside the virtual text to the value of your application's version. So, you would get "My Application 1.0" (or whatever your version is).

Arguments in Virtual Text

Some virtual text fields can also receive arguments that modify the value of the virtual text. In this case, the virtual text will not be a single word surrounded by `<%` and `%>`, but will instead be the first word of the virtual text along with arguments passed after the word. An example is the `<%Date%>` virtual text which can be passed with an argument that specifies the format of the date string. Example:

```
<%Date %m/%d/%Y%>
```

This tells InstallJammer to return today's date in format of Month/Day/Four-digit-year. `%m/%d/%Y` is an argument passed to the Date virtual text.

Another example:

```
<%Dir C:\foo\bar forwardslash%>
```

would return

```
C:/foo/bar
```

Boolean Values in Virtual Text

InstallJammer automatically converts any virtual text value that looks like a boolean into a 0 or 1. This is mostly to make dealing with booleans simpler, but it's also so that checkboxes respond accurately to virtual text since checkboxes expect a 0 or 1 for value and not the various boolean values that are acceptable. Acceptable boolean values are: 0, false, no, or off or 1, true, yes or on, or any abbreviated or uppercase spellings.

You should always check virtual text variables that are boolean by checking to see if the string is true or false and not a specific string like yes or no. The [String Is Condition](#) is better than a [String Equal Condition](#) for boolean values.

-0-

Virtual Text Definitions

The following virtual text definitions are currently available in InstallJammer. Anytime you see ?something? inside of a virtual text field, it means that that virtual text can or must have an argument passed to it.

<% AllowLanguageSelection% >

If true, the user will be prompted to select a language at the beginning of the install.

<% AppName% >

The name of the application as set during build time in the Application Information tab.

<% ApplicationID% >

A unique identifier for this application. This ID is set when the install project is first created and never changes through the life of the project. It can be used to identify previous installs of the same application on a target system, and InstallJammer uses it during uninstallation.

<% BuildVersion% >

The build version is the fourth value of the <% InstallVersion%>.

<% CancelledInstallAction% >

Contains the text describing the action InstallJammer will take if the install is cancelled mid-process. This virtual text is defined under the Install Features in the Application Information tab.

<% CleanupCancelledInstall% >

Tells InstallJammer whether it needs to cleanup a cancelled install or not.

<% Company% >

The company name set during build time in the Application Information tab.

<% ConsoleMode% >

True if the install is currently being run in console mode.

<% CurrentAction% >

The ID of the current action being executed.

<% CurrentCondition% >

The ID of the current condition being executed.

<% CurrentObject% >

The ID of the current object being used in the system. This can be a pane, action or condition.

<% CurrentPane% >

The Pane ID of the current pane displayed to the user.

<% Date ?dateFormat?% >

A formatted string of today's date based on <% dateFormat%>. The Date virtual text can receive an optional argument of the format to use for the date. Defaults to <% Date <% dateFormat%>%>

<% DateFormat% >

The default format to use for <% Date%> when no format argument is specified. Defaults to %Y%m%d, which looks like 20051201.

<% Debugging% >

If true, InstallJammer will run in debugging mode. Debug mode outputs debug messages to a log file and saves the temporary directory used during installation for later debugging.

<%DefaultLanguage% >

The default language if none is specified or if the user is not allowed to choose a language.

<%DefaultMode% >

True if the user selected to run in the install in Default mode.

<%Dir ?directory or filename? ?platform?% >

Take the given directory and normalize it so that it matches the format of the current platform. On Windows this will use a \ as the path separator. On UNIX, the path separator will be / and any drive lettering (C:, D:, etc...) will be stripped off.

If a second argument is specified, it can be one of the following: backslash, forwardslash, platform, unix or windows. Backslash and forwardslash specifies which path separator to use but no other modifications are done to the directory. Unix or Windows will behave as described above for the given platform.

<%Dirname ?directory or filename?% >

Given a directory or filename, return the parent directory. Example: <%Dirname C:/foo/bar%> would return C:/foo.

<%DiskSpace ?disk space?% >

Returns a pretty text string describing the space in disk space. Depending on the size, this will be N KB, N MB, or N GB where N is the size.

<%DOSName ?directory or filename?% >

Given a directory or filename, return the 8.3 short DOS name on Windows. This will return something like C:/Progra~1/file by using the Windows native short name.

<%Env ?variable?% >

This will return the value of an environment variable if it exists. <%Env PATH%> would return the user's current PATH variable.

<%ErrorDirs% > (set by Uninstall Selected Files)

If <%ErrorsOccurred%> is true, this virtual text may contain a list of directories that the uninstaller failed to remove.

<%ErrorFiles% > (set by Uninstall Selected Files)

If <%ErrorsOccurred%> is true, this virtual text may contain a list of files that the uninstaller failed to remove.

<%Errors% > (set by Uninstall Selected Files)

If <%ErrorsOccurred%> is true, this virtual text will contain a string representation of the errors that occurred during an uninstall. This is usually displayed to the user in the uninstall details.

<%ErrorsOccurred% > (set by Uninstall Selected Files)

This is set during the Uninstall Selected Files action. It will be false if no errors occurred during the uninstall, or it will be true if some error happened while doing the uninstall.

<%Ext% >

Return the executable extension for the current platform. This is .exe on Windows and empty on UNIX.

<%ExtractSolidArchivesOnStartup% >

True if the developer has opted to extract any solid archives when the installer first starts.

<%FileBeingInstalled% >

The file currently being installed. This is only valid during file installation.

<%FileBeingInstalledText% >

A pretty text describing the current file being installed.

Defaults to "Copying <%FileBeingInstalled%>"

<%FileGroup ?file group?% >

Given the name of a file group, return the full directory path for that file group.

<%GroupBeingInstalled% >

The name of the current file group being installed. This is only valid during file installation.

<%GroupBeingInstalledText% >

A pretty text describing the current group being installed.

Defaults to "Installing <%GroupBeingInstalled%>"

<%GUID% >

Generates a new globally unique identifier (GUID).

<%GuiMode% >

True if the install is currently running in a mode that requires a GUI.

<%IncludeDebugging% >

Whether or not debugging options were included in this installer.

<%InstallDirSuffix% >

If this virtual text variable exists in your project, its value will automatically be appended to the <%InstallDir%> anytime it is changed if the <%InstallDir%>'s tail directory does not already match the value. So, if InstallDirSuffix were set to <%AppName%>, for example, the installer would always append the name of your application to the installation directory path no matter where the user chooses to install, unless the InstallDir already has <%AppName%> at the end.

<%InstallDrive% >

This Windows-only virtual text holds the drive letter that the installation is targeted for. So, if <%InstallDir%> is C:/install, <%InstallDrive%> would be C:. This virtual text is automatically updated anytime the <%InstallDir%> value is changed.

<%InstallFinished% >

Whether or not InstallJammer completed the file installation. This is only true if file installation was begun and completed without the user cancelling.

<%InstallHasSolidArchives% >

True if the installer has any solid archives.

<%InstallID% >

A unique identifier for this instance of the running installer. This ID is set when the installer first starts up, and a new one is generated every time the installer is run.

<%InstallMode% >

The current mode of installation chosen by the user. Can be: Console, Default, Silent or Standard.

<%InstallPassword% >

This virtual text is used, by default, in the Install Password pane of most themes. It is empty by default, and will contain a password if the user enters one. This is usually matched against the <%InstallPasswordEncrypted%> virtual text to see if the password is correct.

<%InstallPasswordEncrypted% >

If the project was defined with an install password for the installer, this virtual text will

contain the encrypted form of that password. The plain-text form of the install password is never stored in the installer itself. This virtual text is usually compared against a password the user enters using the Password Test Condition to see if they match.

<%InstallPercentComplete% >

The current percentage complete of the file installation. This is only valid during file installation.

<%InstallRegistryInfo% >

If this virtual text is set to false in your project, InstallJammer will not attempt to create a registry directory separate from your installation to hold its install information. Instead, all of the install information will be stored inside the uninstaller so that it can properly uninstall.

<%InstallSource% >

The full path to the source directory the installation executable was started from.

<%InstallStarted% >

True if the file installation portion of the install was started, whether it completed or not.

<%InstallStopped% >

True if the user cancelled the file installation before it had a chance to complete.

<%InstallType% >

The setup type of installation the user has chosen to install. If the user is not given the option to select a setup type, or that pane has not been shown yet, this variable will default to the Default Setup Type set during build time.

<%InstallVersion% >

The install version of your application. This version is represented as X.X.X.X and is also used by InstallJammer when determining whether to install files based on version numbers.

<%Installer% >

The full path to the installation executable when it was started.

<%InstallerID% >

A unique identifier for this installer. This ID is set when the installer is built and remains the same no matter how many times the install is run. This ID changes with each build of an installer.

<%Installing% >

True if the file installation is currently in progress.

<%Language% >

The current language of the installer as expressed as a language code, an optional country code, and an optional system-specific code, each separated by _. The country and language code are specified in standards ISO-639 and ISO-3166. For example, the locale "en" specified English, and "en_US" specifies U.S. english.

<%LastGUID% >

Contains the value of the last GUID generated by using the <%GUID%> virtual text.

<%LastUUID% >

Contains the value of the last UUID generated by using the <%UUID%> virtual text.

<%LicenseAccepted% >

True if the user has seen and accepted the license agreement for the application.

<%MajorVersion% >

The major version is the first value of the <%InstallVersion%>.

<%MinorVersion% >

The minor version is the second value of the <%InstallVersion%>.

<%PackageDescription% >

The package description defined in the project.

<%PackageLicense% >

The package license defined in the project.

<%PackageMaintainer% >

The package maintainer defined in the project.

<%PackageName% >

The package name defined in the project.

<%PackagePackager% >

The package packager defined in the project.

<%PackageRelease% >

The package release version defined in the project.

<%PackageSummary% >

The package summary defined in the project.

<%PackageVersion% >

The package version defined in the project.

<%PatchVersion% >

The patch version is the third value of the <%InstallVersion%>.

<%PathSeparator% >

The character used to separate paths in variables on the target system. This is ; for Windows and : for UNIX.

<%Platform% >

The name of the current platform being installed. This variable only exists during installation time.

<%ProgramExecutable% >

If specified by the user during build time, this variable contains the path to the executable that will start the installed program upon completion of installation.

<%ProgramFolderName% >

The name of the Program Folder the user has chosen. If the user is not given the option to select the program folder, or that pane has not been shown yet, this variable will default to the Default Program Folder set during build time.

<%ProgramLicense% >

The location of the application's LICENSE file as specified in the project.

<%ProgramReadme% >

The location of the application's README file as specified in the project.

<%PromptForRoot% >

True if the developer has opted to prompt for the root password when a root password is required to run the (un)install.

<%Property ?object? ?property?% >

This will return the value of a property on a given object. If object is specified, the property value for that object will be returned. If no object is specified, the property will be returned

for the currently-active object. This can be whatever action or condition is currently being executed.

`<% RealUsername% >`

The name of the real user running the install. If the user is running as root (through su or sudo), the `<% Username% >` will show root instead of the real user name. This value always reflects the real user.

`<% RegValue ?registry key? ?value?% >`

The current value of a registry key on Windows. If only a key is passed, the default value for that key will be returned. If value is passed, that value within the specified key will be returned.

`<% RequireRoot% >`

True if the (un)installer requires root to run.

`<% RunningInstaller% >`

True if the currently-running program is the installer.

`<% RunningUninstaller% >`

True if the currently-running program is the uninstaller.

`<% ScriptExt% >`

This is the file extension for common script on each platform. This is set to .bat on Windows and .sh on UNIX platforms.

`<% SelectedComponents% >`

A list of components that are currently selected for installation. This should be treated as a Tcl list.

`<% SelectedFileGroups% >`

A list of file groups that are currently selected for installation. This should be treated as a Tcl list.

`<% ShortAppName% >`

A shortened version of your application name. This is usually used on UNIX systems as a directory for your application, so it should usually be all lowercase and contain no spaces.

`<% ShowConsole% >`

If true, InstallJammer will show a debugging console on startup.

`<% SilentMode% >`

True if the user selected to run the install in Silent mode.

`<% SolidArchivesExtracted% >`

True if the installer has any solid archives that have been extracted already.

`<% SpaceAvailable ?directory?% >`

Returns the disk space available on the current drive. If directory is specified, returns the disk space available for that drive.

`<% SpaceAvailableText% >`

Returns a pretty text description of the disk space available on the current drive. Equivalent to `<% DiskSpace <% SpaceAvailable% >% >`.

`<% SpaceRequired% >`

The current amount of space required on the target system to install the selected components.

`<% SpaceRequiredText% >`

A pretty text describing the amount of space required to install.
Equivalent to `<%DiskSpace <%SpaceRequired%>%>`.

`<%Status% >`

This variable is used as the current status of what InstallJammer is doing. During file installation, this is usually set to `<%FileBeingInstalledText%>`, but it can reflect many other actions InstallJammer is taking.

`<%Tail ?directory or filename?% >`

Given a directory or filename, return the tail end of the path. Example: `<%Tail C:/foo/bar%>` would return bar.

`<%Testing% >`

If true, InstallJammer will run in test mode which means no files will be installed or uninstalled. This does not stop other actions that create shortcuts, etc...

`<%UninstallMode% >`

The current uninstall mode: Console, Silent or Standard.

`<%UninstallPercentComplete% >`

The current percentage complete of the file uninstallation. This is only valid during uninstall.

`<%UpgradeApplicationID% >`

The Upgrade Application ID specified by the developer.

`<%UpgradeInstall% >`

If Upgrade Application ID is not null, this value will be true, telling InstallJammer that this is an upgrade install and certain actions should not be performed.

`<%UserIsRoot% >`

True if the user currently installing is root.

`<%UserMovedBack% >`

True if the user has reached the current pane by clicking the Back button.

`<%UserMovedNext% >`

True if the user has reached the current pane by clicking the Next button.

`<%Username% >`

The username of the current system user executing the install. If the user is currently root, this value will show 'root'. If you want to know the user who is currently running as root, you can use `<%RealUsername%>`

`<%Version% >`

The version set during build time in the Application Information tab.

`<%WindowsPlatform% >`

A string specifying the exact version of Windows currently being installed on. Possible versions are: Win95, Win98, WinNT, WinME, Win2k, WinXP, Win2003, Vista, and Windows7.

`<%WizardCancelled% >`

True if the wizard started and the user cancelled the wizard before it completed.

`<%WizardFinished% >`

True if the wizard was started and actually completed successfully without the user cancelling.

`<%WizardFirstStep% >`

True if the pane currently displayed is the first step of the wizard.

`<%WizardLastStep% >`

True if the pane currently displayed is the last step of the wizard.

<% WizardStarted% >

True if the wizard started up successfully. This virtual text will always be false for silent and console installs since no wizard is ever started.

-o-

Virtual Directory Definitions

The following Virtual Directories are currently available in InstallJammer.

<%Desktop% >

The directory location of the user's desktop. On Windows, this is the user's desktop. On UNIX, the value is dependant upon which window manager the user is running during installation.

<%GnomeDesktop% >

The Gnome desktop directory. This is where Gnome stores all of the shortcuts that appear on the desktop. This directory is empty if the user does not have the Gnome Desktop.

<%Home% >

The user's home directory. On Windows, InstallJammer attempts to find the best possible option for the user's home directory if they have not specified a location for it.

<%InstallDir% >

The destination directory the user has chosen to install to. If the user is not given the option to select a destination directory, or that pane has not been shown yet, this variable will default to the Default Destination Directory set during build time.

<%InstallInfoDir% >

The directory where InstallJammer will store all of its registry information for this install or where it will check for registry information during uninstallation.

<%InstallJammerRegistryDir% >

The root directory that InstallJammer is using to store registry information about your installs. This does not include the subdirectory for the current install. This is the directory where the directories for each install are stored.

<%InstallSource% >

The directory the installer executable was in when it was started.

<%KDEDesktop% >

The KDE desktop directory. This is where KDE stores all of the shortcuts that appear on the desktop. This directory is empty if the user does not have the KDE Desktop.

<%ProgramFolder% >

The full path to the Program Folder the user has selected. This is usually a combination of the Programs folder and ProgramFolderName.

<%RootInstallDir% >

The root install dir. If this is not null, and the user is root, <%InstallDir%> will be set to this value on startup.

<%Temp% >

The temporary directory used by InstallJammer during installation. Once InstallJammer has finished, this directory, and all of its contents, will be deleted. This is useful for creating temporary files during installation.

<%TempRoot% >

The root of the temporary directory being used by InstallJammer. InstallJammer will always create a subdirectory in <%TempRoot%> to do all of its work in, but <%TempRoot%> is the actual temporary directory on the user's system. The user can specify this directory on the command-line with the --temp option.



Temp Root is determined by looking at:

TEMP environment variable
TMP environment variable
C:/Documents and Settings/<user>/Local Settings/Temp (Windows only)
C:/Windows/Temp (Windows only)
C:/Winnt/Tmp (Windows only)
C:/Temp (Windows only)
/usr/tmp (UNIX only)
/tmp (UNIX only)
/var/tmp (UNIX only)

Windows Virtual Directories

The following Virtual Directories are also available on the Windows platform. Virtual text in all uppercase letters is treated specially by InstallJammer as a Windows directory.

<%ADMINTOOLS%>

<%ALTSTARTUP%>

File system directory that corresponds to the user's nonlocalized Startup program group.

<%APPDATA%>

File system directory that serves as a common repository for application-specific data. A typical path is C:\Documents and Settings\username\Application Data.

<%RECYCLEBIN%>

Virtual folder containing the objects in the user's Recycle Bin.

<%CDBURN_AREA%>

<%COMMONADMINTOOLS%>

<%COMMONALTSTARTUP%>

File system directory that corresponds to the nonlocalized Startup program group for all users. Valid only for Microsoft Windows NT® systems.

<%COMMONAPPDATA%>

<%COMMONDESKTOPDIRECTORY%>

File system directory that contains files and folders that appear on the desktop for all users. A typical path is C:\Documents and Settings\All Users\Desktop. Valid only for Windows NT systems.

<%COMMONDOCUMENTS%>

<%COMMONFAVORITES%>

File system directory that serves as a common repository for all users' favorite items. Valid only for Windows NT systems.

<%COMMONMUSIC%>

<%COMMONOEMLINKS%>

<%COMMONPICTURES%>

<%COMMONPROGRAMS%>

File system directory that contains the directories for the common program groups that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs. Valid only for Windows NT systems.

<%COMMON_QUICK_LAUNCH% >

The quick launch directory common to all users on the system. Adding a shortcut here will cause it to appear on every user's quick launch bar.

<%COMMON_STARTMENU% >

File system directory that contains the programs and folders that appear on the Start menu for all users. A typical path is C:\Documents and Settings\All Users\Start Menu. Valid only for Windows NT systems.

<%COMMON_STARTUP% >

File system directory that contains the programs that appear in the Startup folder for all users. A typical path is C:\Documents and Settings\All Users\Start Menu\Programs\Startup. Valid only for Windows NT systems.

<%COMMON_TEMPLATES% >

<%COMMON_VIDEO% >

<%COMPUTERSNEARME% >

<%CONNECTIONS% >

<%CONTROLS% >

Virtual folder containing icons for the Control Panel applications.

<%COOKIES% >

File system directory that serves as a common repository for Internet cookies. A typical path is C:\Documents and Settings\username\Cookies.

<%DESKTOP% >

Windows Desktop virtual folder that is the root of the namespace.

<%DESKTOPDIRECTORY% >

File system directory used to physically store file objects on the desktop (not to be confused with the desktop folder itself). A typical path is C:\Documents and Settings\username\Desktop.

<%DRIVES% >

My Computer -- virtual folder containing everything on the local computer: storage devices, printers, and Control Panel. The folder may also contain mapped network drives.

<%FAVORITES% >

File system directory that serves as a common repository for the user's favorite items. A typical path is C:\Documents and Settings\username\Favorites.

<%FONTS% >

Virtual folder containing fonts. A typical path is C:\Windows\Fonts.

<%HISTORY% >

File system directory that serves as a common repository for Internet history items.

<%INTERNET% >

Virtual folder representing the Internet.

<%INTERNET_CACHE% >

File system directory that serves as a common repository for temporary Internet files. A typical path is C:\Documents and Settings\username\Temporary Internet Files.

<%LOCAL_APPDATA% >

<%MYDOCUMENTS%>

<%MYMUSIC%>

<%MYPICTURES%>

<%MYVIDEO%>

<%NETHOOD%>

A file system folder containing the link objects that may exist in the My Network Places virtual folder. It is not the same as NETWORK, which represents the network namespace root. A typical path is C:\Documents and Settings\username\NetHood.

<%NETWORK%>

Network Neighborhood -- virtual folder representing the root of the network namespace hierarchy.

<%PERSONAL%>

File system directory that serves as a common repository for documents. A typical path is C:\Documents and Settings\username\My Documents. This should be distinguished from the virtual My Documents folder in the namespace.

<%PRINTERS%>

Virtual folder containing installed printers.

<%PRINTHOOD%>

File system directory that contains the link objects that may exist in the Printers virtual folder. A typical path is C:\Documents and Settings\username\PrintHood.

<%PROFILE%>

<%PROGRAM_FILES%>

A folder for components that are shared across applications. A typical path is C:\Program Files.

<%PROGRAM_FILESX86%>

<%PROGRAM_FILES_COMMON%>

<%PROGRAM_FILES_COMMONX86%>

<%PROGRAMS%>

File system directory that contains the user's program groups (which are also file system directories). A typical path is C:\Documents and Settings\username\Start Menu\Programs.

<%QUICK_LAUNCH%>

File system directory that contains Windows shortcuts that appear on the user's Quick Launch toolbar. Not valid on Windows 95.

<%RECENT%>

File system directory that contains the user's most recently used documents. A typical path is C:\Documents and Settings\username\Recent.

<%RESOURCES%>

<%RESOURCES_LOCALIZED%>

<%SENDTO%>

File system directory that contains Send To menu items. A typical path is C:\Documents and

Settings\username\SendTo.

<% STARTMENU% >

File system directory containing Start menu items. A typical path is C:\Documents and Settings\username\Start Menu.

<% STARTUP% >

File system directory that corresponds to the user's Startup program group. The system starts these programs whenever any user logs onto Windows NT or starts Windows 95. A typical path is C:\Documents and Settings\username\Start Menu\Programs\Startup.

<% SYSTEM% >

System folder. A typical path is C:\WINNT\SYSTEM.

<% SYSTEM32% >

System folder. A typical path is C:\WINNT\SYSTEM32.

<% SYSTEMX86% >

<% TEMPLATES% >

File system directory that serves as a common repository for document templates.

<% WINDOWS% >

Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment variables. A typical path is C:\Windows.

-0-

Developer's Guide

Debugging an Install

InstallJammer provides several different ways and helpful tools to help debug installs. Most installs are fairly simple and don't really require complex debugging, but some installs can get very elaborate and require the ability to peek into what InstallJammer is really doing in the background.

Commands for debugging

InstallJammer provides two commands to assist with debugging: `debug` and `debugging`. The `debug` command will output a debug message and can be used in scripts or in the console to output debugging information. The `debugging` command is the command that controls what debugging is taking place during installation.

The following is the usage for the `debugging` command:

`debugging level <0, 1, 2 or 3>`

This command sets the level of debugging output to show. 0 means no output, and 3 means very verbose output. The default level when not specified is 1.

`debugging <on | off>`

This command will simply turn debugging on or off.

`debugging <file | console> <on | off> ?filename?`

This command will turn debugging on or off specifically for outputting debug information to a file or to the console. If a filename is given, file output will be directed to that file. If no filename is given, and debug output to a file is turned on, the file `<%Temp%>/debug.log` will be used.

In any case, when you use the `debugging` command, it will tell you exactly what and how it is logging. Calling the `debugging` command with no arguments will tell you the current state of debugging in the installer.



Debug output to the console is controlled by the `::debug` variable.

Debug output to a file is controlled by the `::info(Debugging)` variable.

The `::debugfp` variable is the open file pointer to the debug log file if it is open. The variable will be an empty string if file logging is turned off.

-0-

Builder API

What is the Builder API ?

The Builder API is a set of functions that can be used in control scripts when executing the builder from the command line. This allows you to create scripts that can be passed on the command line that modify your installer based on criteria.

Calling a Builder API proc

All of the procs in the Builder API are in the `::BuilderAPI` namespace. To call a proc in the Builder API, simply use the fully-qualified name of the proc in your code. All arguments passed to an Builder API proc are non-positional key-value pairs. That means that it doesn't matter what order you pass the arguments in since each argument is specified as a -option with a value.

For example, let's say we wanted to change the name of the installer produced for the Windows platform in a certain case. You would add something like this to a control script:

```
::BuilderAPI::SetPlatformProperty -platform Windows -property Executable -value "setup.exe"
```

In this example, we call the proc `::BuilderAPI::SetPlatformProperty` with three arguments: -platform, -property and -value. Each property must have a value after it.

Finding the arguments to an API call

All API calls are loaded into the install builder when it loads, so you can easily go into the console (through the Help->Show Debug Console menu) and type a Builder API call to see what it can do. To see the usage for an API call, type the name of the call and the argument -? into the console. Like this:

```
::BuilderAPI::SetPlatformProperty -?
```

This will dump the usage information, including all possible arguments, for the `::BuilderAPI::SetPlatformProperty` API call. Of course, you can always read the documentation for each call as well, but this might be quicker for some people.

-O-

Builder API Calls

GetAction

::BuilderAPI::GetAction

Find an action object by its alias and setup.

Supported Platforms

All

Returns

Object ID or empty string

Options

-alias

The alias of the action to locate.

-setup

Can be one of install or uninstall to look for the given action within the install tree or the uninstall tree.

-o-

GetActionGroup

::BuilderAPI::GetActionGroup

Find an action group object by its alias and setup.

Supported Platforms

All

Returns

Object ID or empty string

Options

-alias

The alias of the action group to locate.

-setup

Can be one of install or uninstall to look for the given action group within the install tree or the uninstall tree.

-o-

ModifyObject

::BuilderAPI::ModifyObject

Mark an object as active or inactive.

Supported Platforms

All

Returns

Empty string

Options

-active

If this property is true, the given object will be made active. If it false, the object will be deactivated.

-object

The alias or object ID of the object to modify.

-o-

SetPlatformProperty

::BuilderAPI::SetPlatformProperty

Set a platform-specific property for a given platform

Supported Platforms

All

Returns

Empty string

Options

-platform

The platform to set the property for. This can be any platform supported in the current installation, or it can be the word "all" to mean all platforms, the word "unix" to mean all unix platforms, or the word "active" to mean all active platforms.

-property

Specifies the property to set for the given platform. The current properties for platforms are: Active, BuildSeparateArchives, DefaultFilePermission, DefaultDirectoryPermission, Executable, FallBackToConsole, FileDescription, InstallDir, InstallMode, InstallType, ProgramName, ProgramReadme, ProgramLicense, ProgramFolderName, ProgramExecutable, ProgramFolderAllUsers, PromptForRoot, RequireAdministrator, RequireRoot, RootInstallDir, UseUncompressedBinaries, WindowsIcon.

-value

The value to set the given property to.

-O-

Install API

What is the Install API?

The Install API is a set of functions that can be used in your installer to create more complex installers than is normally available through the standard actions and conditions. Some of the actions and conditions in InstallJammer even use pieces of the Install API to do the real work.

The Install API is loaded with procs that you can call from [Execute Script](#) actions or [Script Condition](#) conditions that give you program-level access to the inner workings of InstallJammer.



It is important to know that the Install API is built on-the-fly as your installer is packaged. Only procs that you actually use in your installer will be included in the packaged installer. This really only means that you cannot make API calls from a running console unless you are already using that API somewhere in your installer since the API call would not have been built into your installer.

Calling an Install API proc

All of the procs in the Install API are in the `::InstallAPI` namespace. To call a proc in the Install API, simply use the fully-qualified name of the proc in your code. All arguments passed to an Install API proc are non-positional key-value pairs. That means that it doesn't matter what order you pass the arguments in since each argument is specified as a -option with a value.

For example, let's say we had an Execute Script action where we wanted to do some checking and then set a virtual text variable. It might look something like this:

```
if {![::InstallAPI::VirtualTextExists -virtualtext Foo]} {  
    ::InstallAPI::SetVirtualText -virtualtext Foo -value Bar  
}
```

As you can see from the example, we make two calls to procs in the `::InstallAPI` namespace. This code uses `::InstallAPI::VirtualTextExists` to check and see if the virtual text Foo exists, and if it doesn't already exist, we're going to use `::InstallAPI::SetVirtualText` to set it to Bar.

The arguments `-virtualtext Foo` passed to `::InstallAPI::VirtualTextExists` tells the call that we want to check if the virtual text variable Foo exists. The arguments `-virtualtext Foo -value Bar` tell `::InstallAPI::SetVirtualText` that we want to set the virtual text variable Foo to the value Bar.

Finding the arguments to an API call

All API calls are loaded into the install builder when it loads, so you can easily go into the console (through the Help->Show Debug Console menu) and type an Install API call to see what it can do. To see the usage for an API call, type the name of the call and the argument `-?` into the console. Like this:

```
::InstallAPI::SetVirtualText -?
```

This will dump the usage information, including all possible arguments, for the `::InstallAPI::SetVirtualText` API call. Of course, you can always read the documentation for each call as well, but this might be quicker for some people.

-o-

Install API Calls

AddInstallInfo

::InstallAPI::AddInstallInfo

Add information to be stored in the install log for this installation. InstallJammer automatically creates a log file for every installation that contains a default set of values for each installation. This API lets you add information to that log so that it can be fetched on subsequent installs.

Supported Platforms

All

Returns

Empty String

Options

-key <key> (required)

The key to store the value under.

-value <value> (required)

The value to store with the key.

Example

The following example adds the Spanish language.

```
::InstallAPI::AddInstallInfo -key Foo -value bar
```

This would add a key "Foo" with the value "bar" to the install log. When using a Check for Previous Install action on subsequent installations, you would get a new virtual text called <%PreviousInstallFoo%> that contains "bar".

-o-

AddLanguage

::InstallAPI::AddLanguage

Add a language and language code to the list of available languages in the current installer.

Supported Platforms

All

Returns

Empty String

Options

-language <language name> (required)

The name of the language as it will be displayed to the user.

-languagecode <language code> (required)

The language code for the given language.

Example

The following example adds the Spanish language.

```
::InstallAPI::AddLanguage -language Spanish -languagecode es
```

-O-

CommandLineAPI

::InstallAPI::CommandLineAPI

This API is used to manipulate and query command line options in an installer or uninstaller.

Supported Platforms

All

Returns

See options

Options

-do

Specifies the action the API should take. Possible values are:

- check Check to see if the option given by -option was passed on the command line.
Returns 1 if the option was passed or 0 if it was not.
- exists Check to see if the option given by -option exists as a possible option.
Returns 1 if the option exists or 0 if it does not.

-option

The command line option to check. Any - or / is stripped from the left side of the option before it is checked.

-o-

ComponentAPI

::InstallAPI::ComponentAPI

This API is used to manipulate and query components within an installer.

Supported Platforms

All

Returns

See options

Options

-active

If this option is specified, the active state of each of the given components will be made active or inactive.

-components

A list of components to act on. The word "all" can be passed to signify that all components in the install should be acted on.

-updateinfo

If this option is true, it tells the installer to update all of the information relating to selected components. The default is 1.

Example

The following example would deactivate all components in the current installer

```
::InstallAPI::ComponentAPI -components all -active 0
```

This example would activate a component called "Component 1"

```
::InstallAPI::ComponentAPI -components [list "Component 1"] -active 1
```

Notice that because -components expects a list of components, we use the [list] command to make a proper list before passing the arguments to the API. In this case, it is a list with a single element.

-0-

ConfigAPI

::InstallAPI::ConfigAPI

This API is used to manipulate and query internal configuration options in an installer or uninstaller.

Supported Platforms
All

Returns
See options

Options

-usenativedirectorydialog

If this option is turned on, the installer will use the native dialog when choosing a directory. This only applies to Windows and OS X where native dialogs exist.

-usenativefiledialog

If this option is turned on, the installer will use the native dialog when choosing a file from the system. This only applies to Windows and OS X where native dialogs exist.

-usenativemessagebox

If this option is turned on, the installer will use the native dialog when displaying messages to the user. This only applies to Windows and OS X where native dialogs exist.

-o-

CopyObject

::InstallAPI::CopyObject

Make a copy of an existing object in the current environment.

Supported Platforms

All

Returns

Name of the new object

Options

-object <object ID or alias> (required)

The Object ID or Alias of the object to be copied.

-newobject <object name>

The name to give to the new object. If this option is not specified, the new object will be created with a unique ID.

-o-

ErrorMessage

::InstallAPI::ErrorMessage

Display an error message to the user using standard tools.

Supported Platforms

All

Returns

Empty String

Options

-message <error message> (required)

This is the body of the message to display to the user in the dialog box or in the console.

-title <dialog title>

This is the window title of the dialog box when it pops up. If not specified, it will default to "Install Error."

-o-

DestroyWidget

::InstallAPI::DestroyWidget

Completely destroys a widget or list of widgets.

Supported Platforms

All

Returns

Empty string

Options

-widgets

A list of widget IDs or aliases. Each widget can be specified by its ID, an alias or in the form of <PANE>.<WIDGET> where <PANE> is the ID or Alias of a pane and <WIDGET> is the ID or Alias of a widget on that pane.

-o-

Exit

::InstallAPI::Exit

Exit the running program as either a canceled or finished install.

Supported Platforms

All

Returns

Empty string

Options

-exitcode <exit code>

If this option is specified, it will set the exit code of the running (un)installer before it exits.

-exittype <cancel | finish>

Tell the installer whether to exit as a canceled install or a finished install. This affects whether the Cancel or Finish actions are executed as the installer exits.

-0-

Fetch URL

::InstallAPI::FetchURL

Fetch the given URL and store its contents into virtual text or in a file.

Supported Platforms
All

Returns
Empty string or error.

Options

-blocksize <size>

The block size to use when downloading. The smaller the block size, the finer the progress bar will be, but it will usually be slower because more calls are made to download the full file size.

-file <filename>

The name of a file to store the downloaded file to. This can also be a list of files separated by ; that corresponds to a list of URLs separated by ;. Each URL relates to one file in the list.

-progressvirtualtext <virtual text>

The name of a virtual text variable (without <% and %>) to update as the progress of the download. This is usually attached to a progress bar.

-proxyhost <hostname>

The host name of a proxy server to use for the request.

-proxyport <port number>

The port number of a proxy server to use for the request.

-statusvirtualtext <virtual text>

The name of a virtual text variable (without <% and %>) to update with status as the files are downloaded. This will set the status virtual text to <%DownloadVirtualText%> as the files are downloaded. If more than one file is specified, the status will also contain the current and total file count.

-url <URL>

A URL or a list of URLs separated by ; to download.

-virtualtext <virtual text>

A virtual text variable or list of virtual text variables separated by ; (without <% and %>) to store the downloaded files into. Each URL relates to a single virtual text variable.

-o-

FindProcesses

::InstallAPI::FindProcesses

Find a list of processes on the target system.



The API uses the system 'ps' command to find processes. The arguments to ps are manipulated based on the version of ps and what options it supports.

Supported Platforms

All

Returns

A list of process IDs (pids) that match the given criteria. Passing no arguments will return a list of all pids. If no processes are found, an empty string is returned.

Options

-glob <yes or no> (Windows only)

If this option is specified, the -name option will be matched as a glob pattern instead of an exact match.

-group <group name or ID>

Find a list of processes running for the given group name or ID. This option does nothing on Windows platforms.

-name <command name>

Find a list of processes that match the given command name. This does not include paths, only the actual command name.

-pid <process id>

Find the explicit process ID given. This is useful if you want to find out if a given process ID is running. If the pid is not found, FindProcesses will return an empty string.

-user <user name or ID>

Find a list of processes running for the given user name or ID.

Example

The following example would find the running process for the gnome-panel.

```
::InstallAPI::FindProcesses -name gnome-panel
```

This example would find all processes being run by the root user.

```
::InstallAPI::FindProcesses -user root
```

-o-

GetComponentsForSetupType

::InstallAPI::GetComponentsForSetupType

Return a list of all the components that are included in the given Setup Type.

Supported Platforms

All

Returns

List of components or an empty list.

Options

-setuptype <setup type name or ID> (required)

The setup type can either be the name of the setup type (like Typical or Custom), or it can be the ID of the setup type. The API call will figure out what you mean.

-O-

GetInstallSize

::InstallAPI::GetInstallSize

Get the required install size for a given object. The object can be a Setup Type, Component or File Group.



With no options at all, this API returns the size that is required for the currently-selected components in the install.

Supported Platforms

All

Returns

The install size required for the given object.

Options

-activeonly <boolean>

If this option is true, only active objects will be added to the total install size. This option is true by default.

-object <object ID or alias>

The object can be a Setup Type, Component or File Group to return the size of. If no object is specified, the install size of the currently-selected options by the user will be returned.

-o-

::InstallAPI::GetSelectedFiles

Get a list of the currently-selected files to be installed. This will return a list of all the files that are currently selected to be installed when InstallJammer reaches the [Install Selected Files](#) action.



The API will automatically exclude any components, file groups or files that are inactive since they would not be installed by Install Selected Files. The selected files are also based on the currently-selected Setup Type.

Supported Platforms

All

Returns

A list of file object IDs or a list of file names depending on options.

Options

-fileids <boolean>

If this option is true, the return value will be a list of file object IDs instead of a list of file names. By default, this API returns a list of the target file names to be installed.

-o-

GetSystemPackageManager

::InstallAPI::GetSystemPackageManager

Returns a string describing the package manager used by the target system.



This API call checks to see if the dpkg or rpm command exists on the system. If a system were to have both dpkg and rpm on the same system, the API will return dpkg since it is the first one checked.

Supported Platforms

UNIX

Returns

DPKG if the system uses the DPKG package database.

RPM if the system uses the RPM package database.

Empty string if neither is found to be used.

-0-

GetWidgetChildren

::InstallAPI::GetWidgetChildren

Return all the children of a given widget.

Supported Platforms

All

Returns

A list of child widgets.

Options

-includeparent <yes | no> (Default: no)

If this option is true, the returned list will also include the given parent widget.

-widget <widget> (required)

The Object ID or alias of the widget whose children are to be found.

-window <window> (Default: Current Pane)

The Object ID or alias of the pane the widget is a child of.

-0-

GetWidgetPath

::InstallAPI::GetWidgetPath

Return the actual widget path of a given widget object or alias.

Supported Platforms

All

Returns

The full path to the widget.

Options

-widget <widget> (required)

The Object ID or alias of the widget whose path is to be returned.

-window <window> (Default: Current Pane)

The Object ID or alias of the pane the widget is a child of.

-o-

EnvironmentVariableExists

`::InstallAPI::EnvironmentVariableExists`

Check to see if a given environment variable exists or not in the current system environment.



This API call just looks into Tcl's `::env()` array to see if the variable exists or not. The `::env()` array is built by Tcl based on the user's environment.

Supported Platforms

All

Returns

True or false.

Options

`-variable <varName>` (required)

The variable to check for existence.

-0-

FindObjects

::InstallAPI::FindObjects

Search the system for an object based on the given options. See [Object Types](#) for the types of objects in a project.

Supported Platforms
All

Returns
A list of matching objects or an empty string.

Options

-active <yes or no>

If specified, only objects which are in the state given will be returned. -active yes means to return only objects which are active, and -active no means to only return objects which are inactive.

-alias <alias name>

The alias of an object to find. If this option is specified, only a single object will be returned since no more than one object can ever have an alias. If no object is found with the given alias, an empty string is returned.

-component <component name>

If specified, only objects that were created from the given component will be included in the result.

-glob <yes or no>

If this option is specified with the -name option, the -name option is matched as a glob pattern instead of an exact name.

-name <name>

If specified, only objects which have the given name will be included in the result. If the -glob option is true, the name will be matched against a pattern instead of an exact match.

-parent <parent ID or alias>

If specified, only objects which are children of the given parent will be included in the result.



Note that while in a project file, files are children of their parent directory, when files are added to an installer, the structures are flattened so that each file is a child of its file group.

-type <object type>

If specified, only objects which are of the given type will be included in the result. Valid types are: file, filegroup, component, setuptype, condition, action, actiongroup and pane.

Example

The following example finds all the files in the Program Files group (if it has the Program Files alias).

```
::InstallAPI::FindObjects -parent "Program Files"
```

The following example finds all the panes.

```
::InstallAPI::FindObjects -type pane
```

The following example finds all the actions in the Install Actions group.

```
::InstallAPI::FindObjects -type action -parent "Install Actions"
```

The following example gets the object ID for the Install Actions group.

```
::InstallAPI::FindObjects -alias "Install Actions"
```

-O-

LanguageAPI

::InstallAPI::LanguageAPI

This API is used to manipulate and query language options in an installer or uninstaller.

Supported Platforms

All

Returns

See options

Options

-do

Specifies the action the API should take. Possible values are:

Set the current language of the installer.

setlanguage

-language

The name or code of a language to use.

-o-

LoadMessageCatalog

::InstallAPI::LoadMessageCatalog

Load a message catalog file into the current installer. This will overwrite text that is already stored in a message catalog or create new message catalogs for languages that do not already exist. Loading messages for a language that does not exist will not add the language to the installer. Use the [AddLanguage](#) API for that.

Supported Platforms

All

Returns

A list of language codes used.

Options

-data <data string>

This data will be parsed just like the contents of a message catalog. This option can be used to load messages that are read from some other location instead of loading from a file.

-dir <directory>

This option specifies a directory to be searched for *.msg files. Each file in the directory must be named after its language code, and each file will be loaded based on the language code and contents of the file. Loading a message catalog for a language that does not exist will not result in an error, but the language will not be added to the running installer. Use the [AddLanguage](#) API for that.

-encoding <encoding>

This option specifies the encoding to use when reading a message catalog from a file or from data.

-file <filename>

The name of a message catalog file to load. If the name of the file is <code>.msg where <code> is a valid language code in the current installer, the message catalog will be loaded into that language. If the file does not match a language in the installer, you must specify the -language option to tell InstallJammer which language to add the messages to.

-language <language or language code>

The language to load the message catalog into. If this option is specified, it will override any other option that InstallJammer might use to determine the language from the file name.

-object <object ID or alias>

This option specifies the ID or alias of a file object contained within the installer that should be loaded as a message catalog. This allows you to package message catalogs into your installer along with the rest of your files and then load them as-needed at runtime. If the name of the file does not match a language in the installer, you must specify the -language option to tell InstallJammer which language to add the messages to.

Example

The following example loads a message catalog packaged within the installer with an alias of Messages File and adds the messages to the English language.

```
::InstallAPI::AddLanguage -language English -object "Messages File"
```

The following example loads any *.msg file from a directory called msgs located in the same directory as the installer. These files would most likely be included with a release on CD. Notice the -subst 1 option that tells the API to substitute the options passed for virtual text.

```
::InstallAPI::AddLanguage -subst 1 -dir <%InstallSource%/msgs
```

The following example loads a single file located in the directory with the installer into the Spanish language.

```
::InstallAPI::AddLanguage -subst 1 -file <%InstallSource%/es.msg
```

-O-

PromptForDirectory

::InstallAPI::PromptForDirectory

Pop up a window prompting the user to choose a directory on the target system.

Supported Platforms

All

Returns

A full directory path if the user selects one or an empty string if they cancel.

Options

-message <message string>

A message to display at the top of the dialog window.

-normalize <boolean>

How to normalize the directory the user chooses. The options are: backslash, forwardslash, platform, unix or windows. Backslash will replace all directory separators with backslashes. Forwardslash will replace all directory separators with forward slashes. Platform means to do whatever the current platform does. Unix means to use forward slashes and remove any Windows drive letter (C:, D:, etc...). Windows means to use back slashes and leave any drive letter in place. The default is platform.

-title <title string>

A title string to display in the title bar of the dialog.

-variable <variable name>

A variable name to store the resulting directory in.

-virtualtext <virtual text name>

A virtual text name (without the <% and %>) to store the resulting directory in.

-0-

PromptForFile

::InstallAPI::PromptForDirectory

Pop up a window prompting the user to choose a file (or multiple files) on the target system.

Supported Platforms

All

Returns

A full directory path if the user selects a file, a list of files if the user is allowed to select multiple files or an empty string if they cancel.

Options

-defaulttextension <file extension>

The default file extension to open the dialog with as well as the default extension to append if the user types in the name of a file without an extension.

-filetypes <file type list>

This is a Tcl list of lists that specifies valid file types for the dialog. The file types are specified as a list of lists where each inner list contains two elements: a text description and a file extension. Example:

```
{ {Text Files} { .txt} }  
{ {All Files} { *} }
```

-initialdir <directory>

The initial directory to start the dialog in.

-initialfile <filename>

The initial file (or files in multiple mode) to select in the dialog when it first pops up.

-message <message string>

A message to display at the top of the dialog window.

-multiple <boolean>

Whether or not the user should be allowed to select multiple files.

-normalize <boolean>

How to normalize the directory the user chooses. The options are: backslash, forwardslash, platform, unix or windows. Backslash will replace all directory separators with backslashes. Forwardslash will replace all directory separators with forward slashes. Platform means to do whatever the current platform does. Unix means to use forward slashes and remove any Windows drive letter (C:, D:, etc...). Windows means to use back slashes and leave any drive letter in place. The default is platform.

-title <title string>

A title string to display in the title bar of the dialog.

-type <open or save>

The type of dialog to use, either: open or save. Open means that the dialog expects the file to already exist on the system to be opened. Save means that dialog can specify a file that does not already exist, and it will prompt the user if the file does exist.

-variable <variable name>

A variable name to store the resulting directory in.

-virtualtext <virtual text name>

A virtual text name (without the <% and %>) to store the resulting directory in.

-o-

PropertyFileAPI

::InstallAPI::PropertyFileAPI

This API provides the necessary functions for reading, manipulating and writing common properties files. A properties file follows the standard convention for Java properties files, but it basically matches an almost universal format for simple properties within a file.

Supported Platforms

All

Returns

See options

Options

-array <arrayName> (required)

The name of an array to store the values in.

-do (required)

Specifies the action the API should take. Possible values are:

append	Add a line to the given array that should be written to the properties file as-is.
data	Return a string of data that will be written to a file if written. This data is the current contents of the properties file.
keys	Return a list of keys in the properties array.
read	Read a given properties file into the array.
set	Set the value of a given key to a value. If the key already exists, it will be overwritten, and when it is saved to a file, it will replace the original line that set the key in the same place within the file.
unset	Unset the value of a given key. This will remove the key from the array as well as delete any reference to the value when the file is written out.
write	Write the property array to a file.

-file

If the do action is read or write, this property specifies the name of the file to read from or write to.

-key

If the do action is set or unset, this property specifies the key in the array to operate on.

-line

If the do action is append, this property specifies a line (or multiline) string to append directly to the given array.

-separator

Specifies the separator character to use between keys and values. The default is =.

-value

If the do action is set or unset, this property specifies the value to set for the given key.

Example

The following example shows a sample properties file and how the API uses it.

```
## Sample .properties file.
```

```
## Set foo  
foo=bar
```

```
## Set bar  
bar=foo
```

```
::InstallAPI::PropertyFileAPI -do read sample.properties -array props  
::InstallAPI::PropertyFileAPI -do set -array props -key foo -value "NEW FOO"  
::InstallAPI::PropertyFileAPI -do append -array props -line "\n## New value"  
::InstallAPI::PropertyFileAPI -do set -array props -key newvalue -value "NEW  
VALUE"
```

The new properties file will look like this

```
## Sample .properties file.
```

```
## Set foo  
foo=NEW FOO
```

```
## Set bar  
bar=foo
```

```
## New value  
newvalue=NEW VALUE
```

-O-

ReadInstallInfo

::InstallAPI::ReadInstallInfo

Read the install information of a previous installation. This API will read information from the given application / install ID and store the values in an array.

Supported Platforms
All

Returns
Empty String

Options

-applicationid <Application ID> (required)

The Application ID to read information for. This can be the installer's own <%ApplicationID%>, or it can be the ID of another, known install project. This lets your installer read the install information of another installer if you know the proper ID.

-array <arrayName> (required)

The name of an array to store the values in.

-installid <Install ID>

The install ID to read information for. If no ID is given, all of the installs for the given application ID will be read and stored in the array in the order they were created.

Example

The following example adds the Spanish language.

```
::InstallAPI::ReadInstallInfo -applicationid SomeID -array tmp  
parray tmp
```

This would load all of the previous installations for the application SomeID into the array tmp, and the parray tmp would dump the array to the console.

-0-

ResponseFileAPI

::InstallAPI::ResponseFileAPI

Manipulate options for response files within installers. This API can be used to read and write response files as well as manipulate the virtual text values that will be saved in an installer response file.

Supported Platforms

All

Returns

Empty string

Options

-do (required)

Specifies the action the API should take. Possible values are:

- add Add the list of virtual text given by -virtualtext to the list of variables to save in the response file.
- read Read the file given by -file as a response file.
- remove Remove the list of virtual text given by -virtualtext from the list of variables to save in the response file.
- write Write the installer response to the file given by -file.

-file

Specifies a filename for read and write operations.

-virtualtext

Specifies a list of virtual text for add and remove operations.

-O-

RollbackInstall

::InstallAPI::RollbackInstall

Rollback any files, directories and registry entries installed on the system during the current installation.

Supported Platforms
All

Returns
Empty string

-o-

SetActiveSetupType

::InstallAPI::SetActiveSetupType

Sets the active Setup Type in the current installer.



This call will set the <%InstallType%> and <%InstallTypeID%> variables in the installer.

Supported Platforms

All

Returns

Empty string

Options

-setuptype <setup type name or ID> (required)

The setup type can either be the name of the setup type (like Typical or Custom), or it can be the ID of the setup type. The API call will figure out what you mean.

-0-

SetExitCode

::InstallAPI::SetExitCode

Set the exit code for the current (un)installer. By default, an (un)installer will exit with an exit code of 0 if it finishes or 1 if it is canceled. By setting the exit code you are forcing an exit code of your choice.

Supported Platforms

All

Returns

Empty string

Options

-exitcode <exit code> (required)

Set the exit code to the given number.

-0-

SetFileTypeEOL

::InstallAPI::SetFileTypeEOL

The the end-of-line character for a given filetype that is to be used when files of that type are installed. This API modifies the behavior of the installer such that each file that matches a given file type will be installed with the specified EOL characters instead of the platform default. This is useful for translating EOL characters from Windows to UNIX or vice-versa when installing files.

Supported Platforms

All

Returns

Empty string

Options

-eol <eol>

The EOL string to use. Can be: auto, binary, crlf, lf, unix or windows. In most cases, this value must be a list of two EOL characters: the character the file is currently using and the character to translate to.

-extension <file extension>

A file extension or a list of file extensions separated by ; that are to be translated with the given EOL.

Example:

This would translate all .txt files that are installed from a Windows (CRLF) EOL to unix (LF) EOL. This assumes that the .txt files stored in your installer already have the Windows EOL.

```
::InstallAPI::SetFileTypeEOL -eol {windows unix} -extension .txt
```

This would translate all .ini files that are installed from a unix (LF) EOL to a Windows (CRLF) EOL.

```
::InstallAPI::SetFileTypeEOL -eol {unix windows} -extension .ini
```

-O-

SetObjectProperty

::InstallAPI::SetObjectProperty

Set the value of a single property on a given object.

Supported Platforms

All

Returns

The new value of the property.

Options

-object <object ID or alias> (required)

The object ID or alias of the object to set the new property on.

-property <property name> (required)

The name of the property to set on the given object.

-value <property value> (required)

The new value to assign to the object property.

Example

The following example would change the destination directory that an Unzip File action is supposed to unzip to.

```
::InstallAPI::SetObjectProperty -object "unzip action" -property Destination  
-value <%InstallDir%/foo
```

-o-

SetVirtualText

::InstallAPI::SetVirtualText

Set the value of a virtual text variable in a given language.

Supported Platforms

All

Returns

The new value of the virtual text.

Options

-action <action ID or alias>

The ID or alias of an action that should be executed anytime the value of the given virtual text changes.

-command <Tcl command>

A Tcl command or proc that should be executed anytime the value of the given virtual text changes.

-language <Language or None>

The language to set the virtual text in. If language is None or not specified, the virtual text will be set without being associated with a language.

-object <object ID or alias>

The ID or alias of an object to set this virtual text on. This option is used to set the text property of a specific object like a pane or action.

-value <new value>

The value to set in the virtual text variable.

-virtualtext <virtual text name> (required)

The virtual text variable name to set (without <% and %>).

Example

The following example would give you a virtual text variable Foo that is set to Bar. This could then be accessed as <%Foo%> from within InstallJammer.

```
::InstallAPI::SetVirtualText -virtualtext Foo -value Bar
```

-0-

SubstVirtualText

::InstallAPI::SubstVirtualText

Given a string, substitute all virtual text out of the string and return the substituted string.

Supported Platforms

All

Returns

The substituted string.

Options

-virtualtext <virtual text string> (required)

The virtual text string to be substituted.

Example

If you had your application name as "My Application" and Version as "1.0", you could do:

```
::InstallAPI::SubstVirtualText -virtualtext "<%AppName%> <%Version%>"
```

which would return: "My Application 1.0"

-O-

URLIsValid

::InstallAPI::URLIsValid

Validate whether a given URL is a valid URL that can be fetched.

Supported Platforms

All

Returns

True or false.

Options

-proxyhost <hostname>

The host name of a proxy server to use for the request.

-proxyport <port number>

The port number of a proxy server to use for the request.

-url <URL>

The URL to validate.

-0-

VirtualTextAPI

::InstallAPI::VirtualTextAPI

This API is used to manipulate and query virtual text in an installer or uninstaller.

Supported Platforms

All

Returns

See options

Options

-do

Specifies the action the API should take. Possible values are:

`settype` Set the type of a virtual text. Type affects how some virtual text is treated within the installer.

-type

Specifies the type to assign to the virtual text. The current available types are: boolean and directory. A boolean type means that the value saved in the virtual text will always be a 0 or 1 based on what is set. A directory type means that the virtual text will automatically be converted to the proper representation for the current platform.

-O-

VirtualTextExists

::InstallAPI::VirtualTextExists

Check to see if the given virtual text variable exists in the given language.

Supported Platforms

All

Returns

True or false.

Options

-language <language or None>

If a language is passed, the API will check to see if the virtual text exists in that language. If no language is passed, or if language is None, the virtual text will be checked in the language-neutral messages.

-virtualtext <virtual text string> (required)

The virtual text string to be substituted.

-o-

Tutorials

Create a New Install Fast

Overview

This tutorial will walk you through the quick and dirty way to get started using InstallJammer with a new install in no time flat. When you're finished with this tutorial, you will have a new install project for your application that is built for the platform you're running the install builder from.

Follow the steps below, and you'll be off and running!

Creating Your First Install

1. Open the New Project Wizard.
2. Type in your Project Name. InstallJammer will fill in the Project Directory based on your Project Name and preferences. Click Next.
3. Fill in your Application Name, Short Application Name, Version and Company. Click Next.
4. Fill in your Install Version and Application Executable fields so that InstallJammer knows how to execute your application after installing. Click Next.
5. Type in the directory where your application is stored on your system. This should be the root directory of your application. Click Finish.
6. Done!

You have just built your install project. Now that you have the project built, click the Build Install button on the toolbar or from the Build menu, and InstallJammer will build an installer for your application.

If you entered your Application Directory properly, you should have your entire application now packaged up in a convenient installer for your current platform!

Click Test Install from the toolbar or from the Build menu and test your new installer!

If you had any trouble, or any questions, please read through [Create a New Install Step-by-Step](#).

-0-

Create a New Install Step-by-Step

Overview

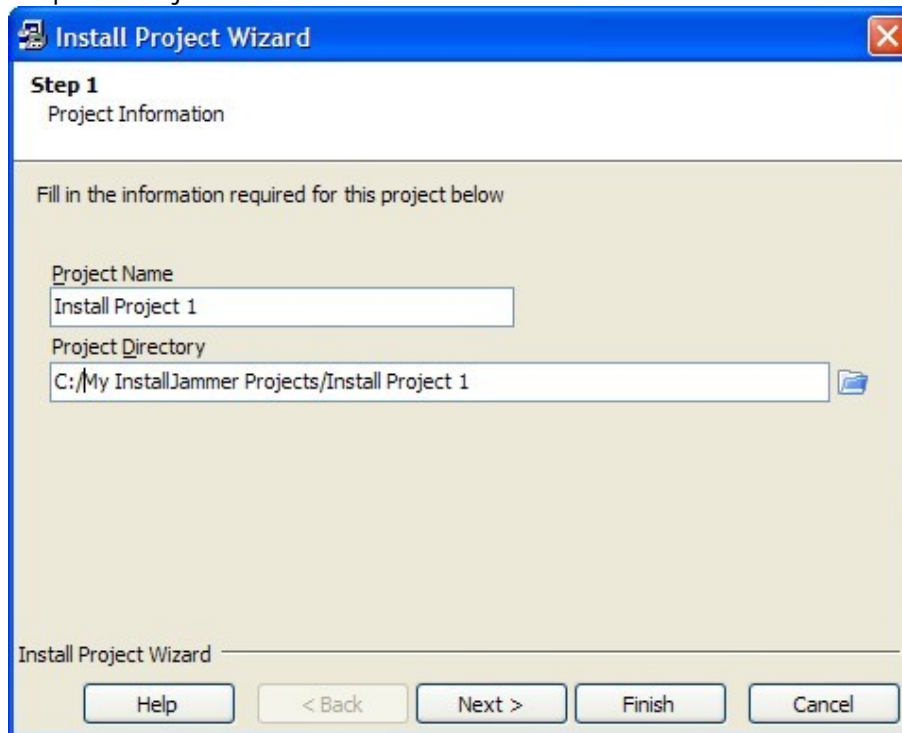
This tutorial will walk you, step-by-step, through creating your first installer using the InstallJammer Install Builder. The process is very easy and should take you no longer than a few minutes. You'll have your first installer up and running in no time!

Creating Your First Install

When InstallJammer first loads, you are presented with the Start Page that shows you a list of current installers you have built. Your first time out, this page will have no projects. So, let's create one!

New installs are created in InstallJammer using the Install Project Wizard. You can open up the wizard by either clicking the "New Install" button on the main toolbar or by opening up the File menu and clicking "New." By clicking either of these options, you should now be presented with the Install Project Wizard. InstallJammer uses the wizard to guide you through the rest of the new install process.

Step 1 - Project Information

The image shows a Windows-style dialog box titled "Install Project Wizard" with a close button in the top right corner. The dialog is divided into two main sections. The top section is titled "Step 1" and "Project Information". Below this, a text label says "Fill in the information required for this project below". There are two input fields: "Project Name" with the text "Install Project 1" entered, and "Project Directory" with the text "C:/My InstallJammer Projects/Install Project 1" entered. A folder icon is visible to the right of the directory field. At the bottom of the dialog, there is a progress bar labeled "Install Project Wizard" and five buttons: "Help", "< Back", "Next >", "Finish", and "Cancel".

Step 1 of the wizard asks you for some basic project information. This information is all internal to InstallJammer and tells it where you want to save your project and what you want to call the project.

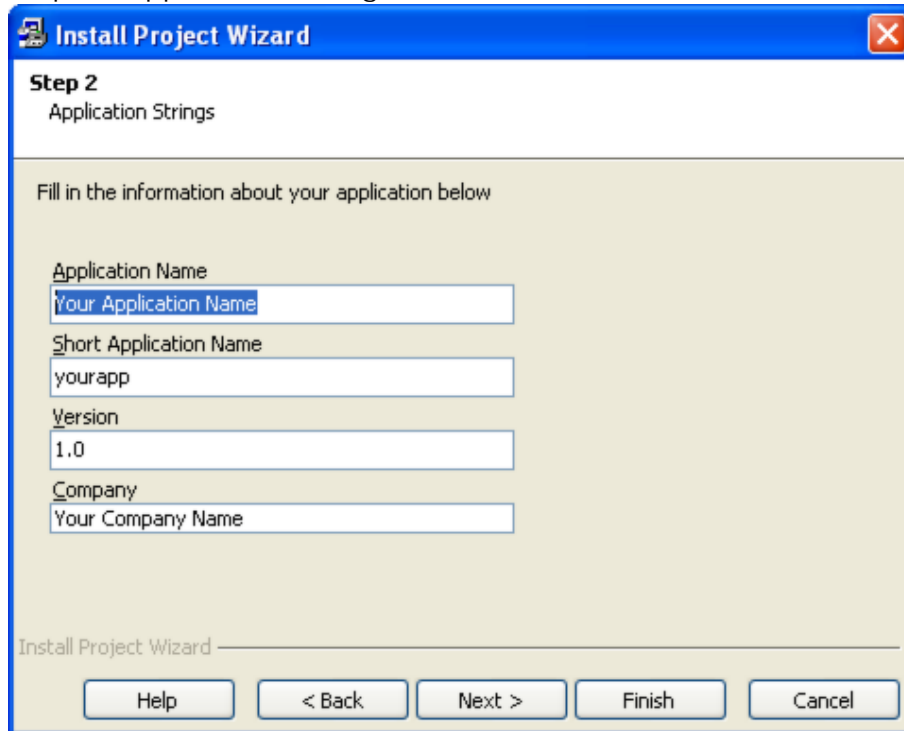
Project Name

This is what you want to call your project. This does not have to be the name of the application you are installing, but it would make sense.

Project Directory

This is the directory where you want to store this project. A new directory will be created to store your install. In InstallJammer, each project is stored in its own directory so that everything can be contained.

Step 2 - Application Strings



The screenshot shows a Windows-style dialog box titled "Install Project Wizard" with a close button (X) in the top right corner. The main title bar is blue. Inside the dialog, the text "Step 2" is bold, and "Application Strings" is below it. A light beige background contains the instruction "Fill in the information about your application below". There are four text input fields: "Application Name" with placeholder text "Your Application Name", "Short Application Name" with placeholder text "yourapp", "Version" with placeholder text "1.0", and "Company" with placeholder text "Your Company Name". At the bottom, there is a progress bar labeled "Install Project Wizard" and five buttons: "Help", "< Back", "Next >", "Finish", and "Cancel".

Step 2 of the wizard asks you some information about the product or program that you will be installing.

Application Name

This is the name of your application or product.

Short Application Name

This is a shortened name for your application. This name is used on most UNIX systems as a directory, so it is usually all lowercase with no spaces.

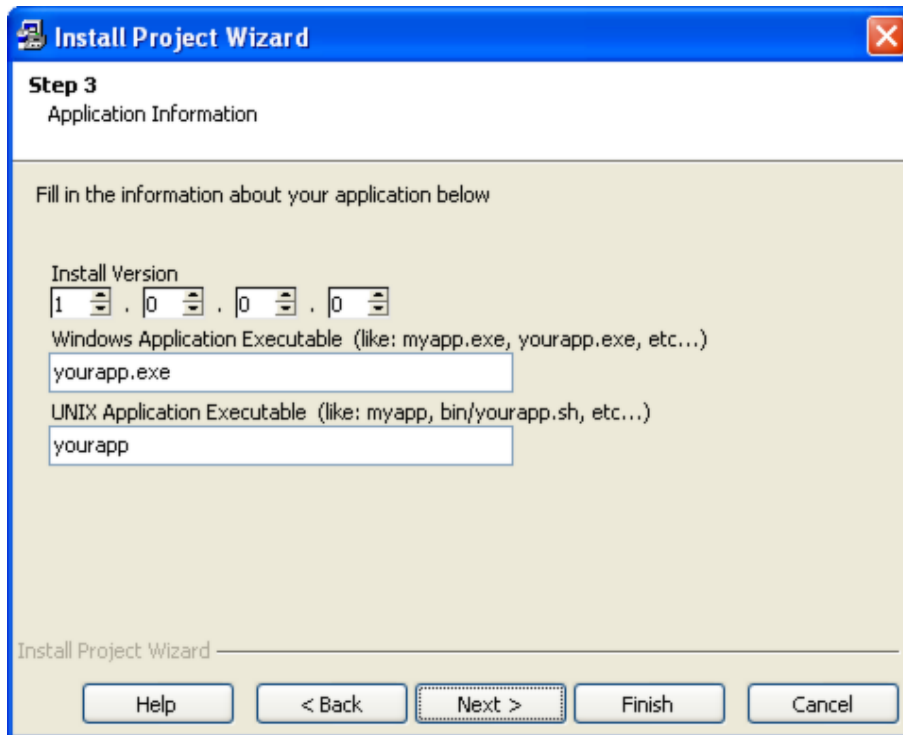
Version

This is the version of your product. This value is not used by InstallJammer for anything, it is there as a representation of your software to the person installing it.

Company

This is the name of your company or organization.

Step 3 - Application Information



Step 3 of the wizard asks for a few more pieces of information about your application.

Install Version

This version is used internally by InstallJammer as a way to track this version of your software against future versions. If you distribute new versions of your software with the version incremented in this field, InstallJammer can (based on your options) install only files that are of the newer version. More on this later.

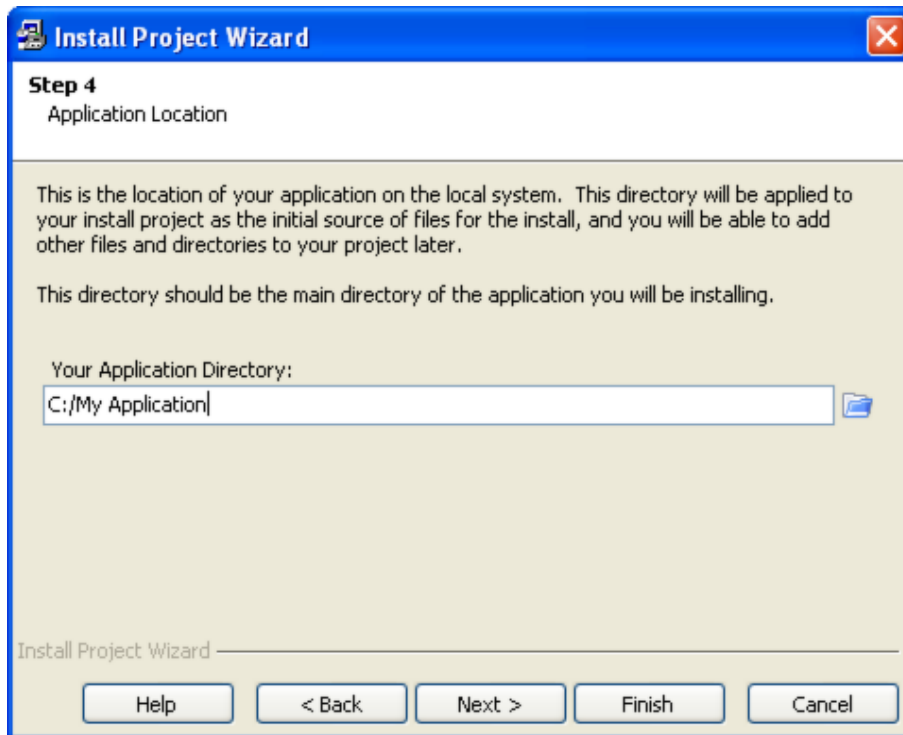
Windows Application Executable

This is the name of the main executable for your application on Windows. This is the entry point into your program, and InstallJammer uses it to make shortcuts and launch your application from within the installer if needed.

UNIX Application Executable

This is the name of the main executable or script for your application in a UNIX environment. This is the entry point into your program, and InstallJammer uses it to make shortcuts and launch your application from within the installer if needed.

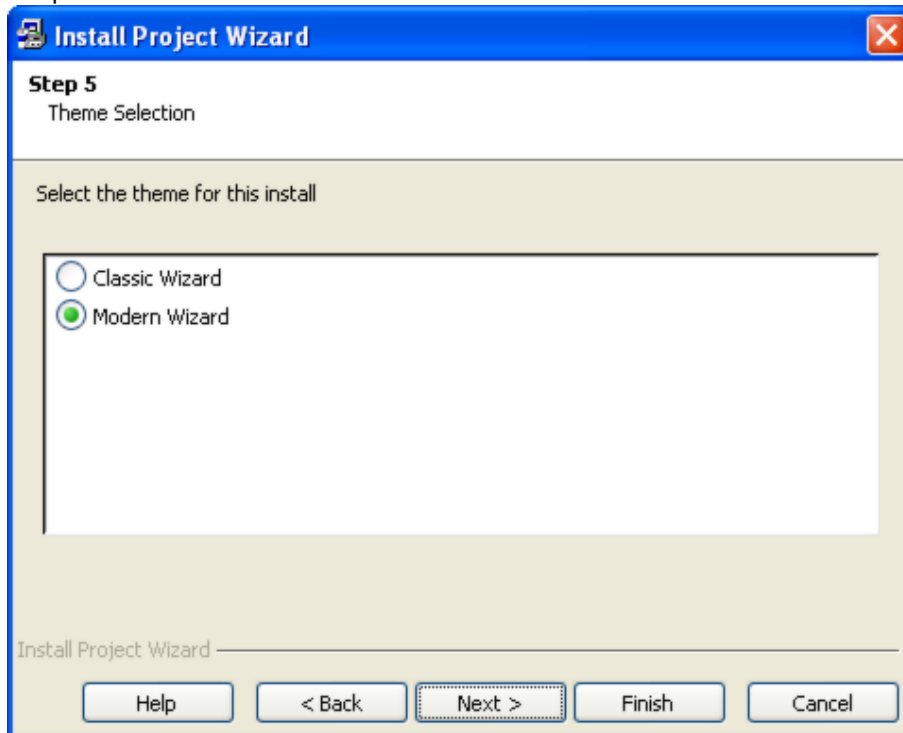
Step 4 - Application Location



Step 4 is a very simple step that asks for the directory that your application is in. By adding your directory here, your application will start with the entire contents of this directory and all of its subdirectories in the project.

You do not have to specify a directory now. You can wait until you get started building the rest of your install and add whatever files and directories you want later.

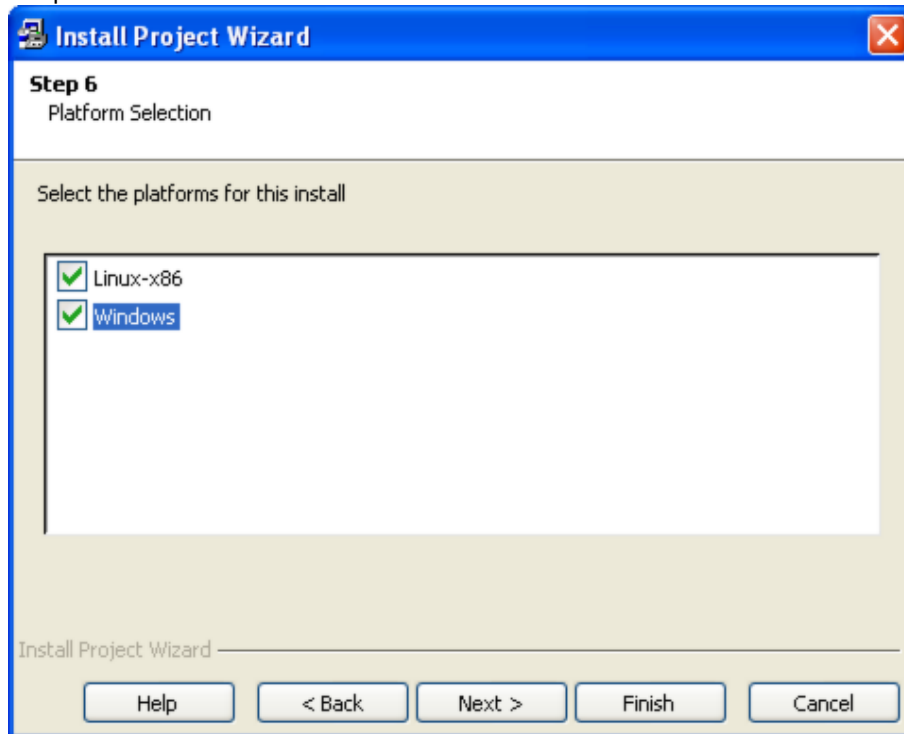
Step 5 - Theme Selection



Step 5 gives you a list of install themes to choose from. The default theme is the Modern

Wizard theme.

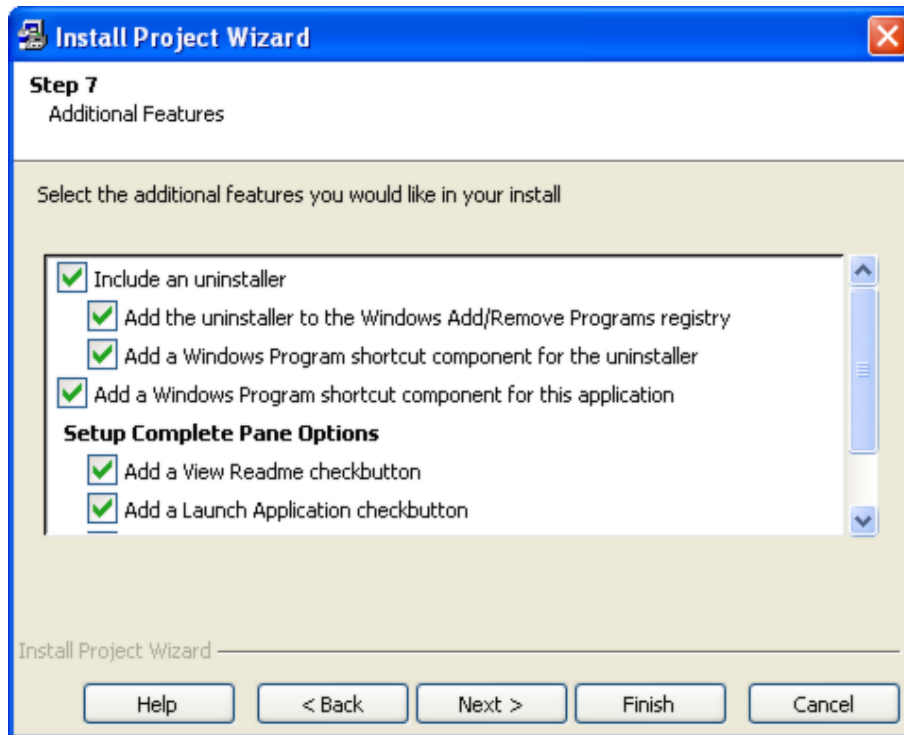
Step 6 - Platform Selection



Step 6 presents you with a list of platforms supported by your installation of InstallJammer and asks you to pick which ones you plan to install on. You need only pick a single platform for your installer if your software is not multiplatform.

By default, only the platform you are building on is selected. Not selecting a platform here does not remove the platform from your installer, it only disables it. This is so that if you decide to add more platforms in the future, you can easily switch them on as needed without having to rebuild your install project.

Step 7 - Additional Features



Step 7 lets you select some additional features you might like to add to your installer. Most of the features are switched on by default, as they are common to most installers used today.

Allow users to select custom components in your install

If this feature is checked, InstallJammer will add two panes to your installer that allow a user to select a Custom install option and then select the custom components they wish to install from your product. This is only important if you have many components in your install and you want the user to be able to choose which parts they install.

Include an uninstaller

If this feature is checked, InstallJammer will add an action to your install actions to create an uninstaller when your application is installed.

Add the uninstaller to the Windows Add/Remove Programs register

If this feature is checked, InstallJammer will add an action to your install actions to create a registry entry in Windows' "Add/Remove Programs" registry. This means your customers will be able to uninstall your program from the main Windows Control Panel.

Add a Windows Program shortcut action for the uninstaller

If this feature is checked, InstallJammer will add an action to your installer to create a Windows shortcut in your application's Program Folder that points to the uninstaller for your application.

Add a Windows Program shortcut action for this application

If this feature is checked, InstallJammer will add an action to your installer to create a Windows shortcut in your application's Program Folder that points to your application. The Program Folder will automatically be created for your application if it does not already exist.

Add a View Readme checkbox

If this feature is checked, InstallJammer will add a View Readme checkbox to the Setup Complete pane of your installer to allow the user to view your README file before exiting the installer.

Add a Launch Application checkbox

If this feature is checked, InstallJammer will add a checkbox to the Setup Complete pane

of your installer to allow the user to launch your application before exiting the installer.

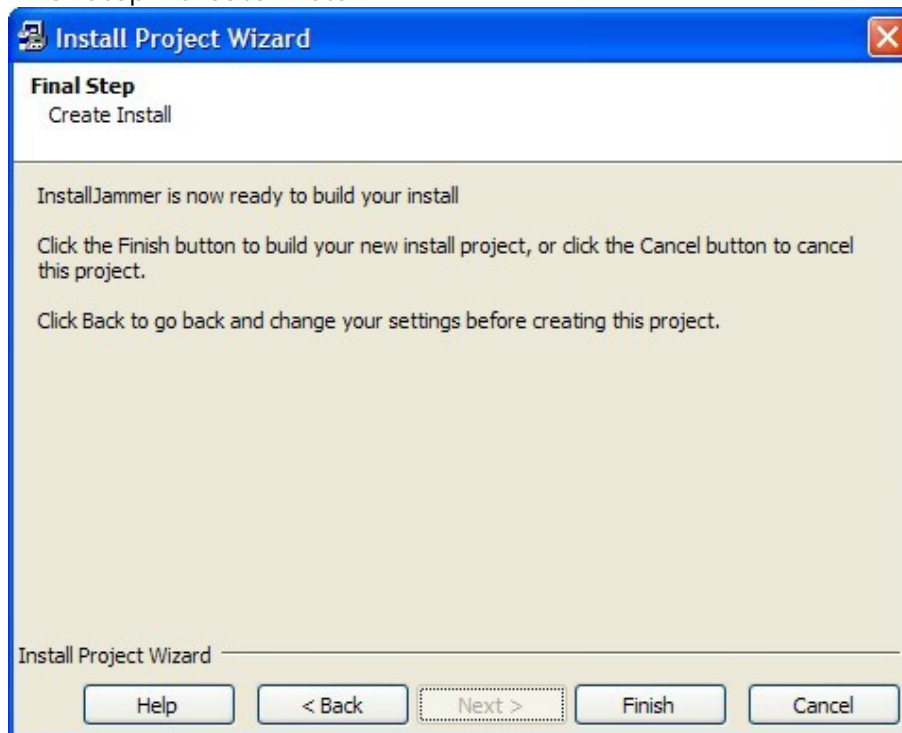
Add a Create Desktop Shortcut checkbox

If this feature is checked, InstallJammer will add a Create Desktop Shortcut checkbox to the Setup Complete pane of your installer to allow the user to create a shortcut to your application on their desktop before exiting the installer.

Add a Create Quick Launch checkbox

If this feature is checked, InstallJammer will add a Create Quick Launch Shortcut checkbox to the Setup Complete pane of your installer to allow the user to create a shortcut to your application in their quick launch toolbar before exiting the installer.

Final Step - Create Install



Now that you have gone through all the steps, you're ready to create your install! Click the finish button, and InstallJammer will create your install and open it to get you started.

Building Your Install

Now you're ready to build your installer! Click the Build Install button on the toolbar or under the Build menu, and InstallJammer will build an installer for your application. If you entered all of your data correctly, you should end up with an installer that is ready to go!

Testing Your Installer

If your installer built successfully, you're ready to give it a test. Click the Test Install button on the toolbar or under the Build menu. Run through the installer, and your program should install in a default location. Make sure you don't change the destination directory when testing your installer. If you do, InstallJammer can't find your installed application to test the uninstaller.

Testing Your Uninstaller

Once you have your application installed, it's time to test your uninstaller. Click the Test Uninstall button on the toolbar or under the Build menu. InstallJammer will execute the uninstaller that was created when your application was installed. Run through the uninstaller to delete the test run you just installed.

-0-

Tcl/Tk Help

What is Tcl/Tk?

Tcl (Tool Command Language) is a very powerful but easy to learn dynamic programming language, suitable for a very wide range of uses, including web and desktop applications, networking, administration, testing and many more. Open source and business-friendly, Tcl is a mature yet evolving language that is truly cross platform, easily deployed and highly extensible.

Tk is a graphical user interface toolkit that takes developing desktop applications to a higher level than conventional approaches. Tk is the standard GUI not only for Tcl, but for many other dynamic languages, and can produce rich, native applications that run unchanged across Windows, Mac OS X, Linux and more.

InstallJammer is written almost entirely in Tcl/Tk with a little bit of help from some extensions written in C. Since all scripting in InstallJammer is done through Tcl/Tk, it might be helpful for you to learn a little about Tcl if you plan on doing any kind of complex operations with your installer. Below are some links to get you started.

[Why would I choose Tcl/Tk over...?](#)

[What does Tcl/Tk code look like?](#)

[A Full Tcl Tutorial](#)

-0-

Index

- A -

- Add Directory to Path 121
- Add Environment Variable 122
- Add Install Info 96
- Add Pane to Order 150
- Add Response File Info 97
- Add to Uninstall 98
- Add Widget 151
- Add Windows File Command 132
- Add Windows File Extension 133
- Add Windows File Type 134
- Add Windows Registry Key 135
- Add Windows Uninstall Entry 136
- AddInstallInfo 209
- Adjust Line Feeds 64
- Append Text to Widget 154
- Application Information 17
- Ask Yes or No 181

- B -

- Backup File 65

- C -

- Change File Ownership 67
- Change File Permissions 68
- Check for Previous Install 99
- Command Line Options 40
- Command Line Test Condition 169
- CommandLineAPI 211
- ComponentAPI 212
- Components 34
- ConfigAPI 213
- Console Ask Yes or No 54
- Console Clear Screen 55
- Console Get User Input 56
- Console Message 57
- Console Pause 58
- Continue Install 79
- Continue Windows Service 142
- Copy File 69
- CopyObject 214
- Create a New Install Fast 249
- Create a New Install Step-by-Step 250
- Create File Link 70
- Create Folder 71
- Create Install Panes 155
- Create Windows Service 143

- D -

- Debugging an Install 201
- Delete Environment Variable 123
- Delete File 72
- Delete Windows Service 146
- Destroy Widget 156
- DestroyWidget 216
- Disable Wow64 Redirection 130
- Disk Builder 44

- E -

- Env Variable Test Condition 178
- EnvironmentVariableExists 226
- ErrorMessage 215
- Execute Action 59
- Execute External Program 60
- Execute Script 63
- Execute Script Condition 170
- Exit 80, 217

- F -

- Fetch URL 81, 218
- File Exists Condition 166
- File Extension Test Condition 182
- File Name Test Condition 167
- File Permission Condition 168
- File Type Test Condition 183
- Files and Directories 31
- FindProcesses 219
- Focus On Widget 157
- Frequently Asked Questions 10

- G -

- Generate UID 82
- Get Java Property 110
- Get Previous Install Info 101
- GetAction 203
- GetActionGroup 204
- GetComponentsForSetupType 220
- GetInstallSize 221
- GetSelectedFiles 222
- GetSystemPackageManager 223
- Getting Started 9
- Getting to Know the Install Builder 12
- GetWidgetChildren 224
- GetWidgetPath 225
- Groups and Files 28

- I -

- Import Windows Registry File 138
- Install API 207
- Install Builder Command Line Options 15

Install Builder Preferences 13
Install Desktop Shortcut 113
Install Log File 103
Install Program Folder Shortcut 115
Install Selected Files 104
Install Uninstaller 105
Install UNIX Program Folder 117
Install UNIX Shortcut 118
Install Windows Shortcut 120
Install Wish Binary 106
Install Wrapped Script 107

- L -

LanguageAPI 229
Launch File 83
Launch Web Browser 84
Locate Java Runtime 111
Log Debug Message 85

- M -

Message Box 86
Message Panel 87
Modify Object 89
Modify Widget 158
ModifyObject 205
Move Forward 159
Move to Pane 160

- O -

Object Test Condition 171
Object Types 47

- P -

Package and Archive Information 26
Package Test Condition 179
Panels and Actions 38
Pause Install 90
Pause Windows Service 147
Platform Condition 174
Platform Information 22
Populate Components 161
Populate Setup Types 162
Port Test Condition 180
PromptForDirectory 232
PromptForFile 233
PropertyFileAPI 235

- R -

Read File Into Virtual Text 73
ReadInstallInfo 237
Reboot or Shutdown System 124
Register Package 125

Register Windows Library 139
Registry Test Condition 184
Remove Directory from Path 127
Remove Pane from Order 163
Remove Windows Registry Key 140
Rename File 74
Replace Text in File 75
ResponseFileAPI 238
Revert Wow64 Redirection 131
RollbackInstall 239

- S -

Script Condition 172
Set Object Property 91
SetActiveSetupType 240
SetExitCode 241
SetFileTypeEOL 242
SetObjectProperty 243
SetPlatformProperty 206
Setup Types 36
SetVirtualText 244
Standard Action Properties 53
Standard Condition Properties 165
Standard Properties 48
Stop Install 93
Stop Windows Service 149
String Equal Condition 175
String Is Condition 176
String Match Condition 177
SubstVirtualText 245

- T -

Test Installer 45
Test Uninstaller 46
Text Window 94

- U -

Uninstall Leftover Files 128
Uninstall Selected Files 129
Unpack Stored File 109
Unregister Windows Library 141
Unzip File 78
URLsValid 246

- V -

Virtual Directory Definitions 196
Virtual Text 43
Virtual Text Definitions 188
Virtual Text Test Condition 173
VirtualTextAPI 247
VirtualTextExists 248

- W -

Wait 95
Welcome 8
What are Action Groups? 52
What are Actions? 51
What are Conditions? 164
What are Panes? 49
What is Tcl/Tk? 258
What is the Builder API? 202
What is the Install API? 207
What is Virtual Text? 186
Write Text to File 76

<http://www.installjammer.com>
