Mt Wilson

**Installation**

# Requirements

## Fully automated installation

The Mt Wilson installation procedure must be fully automated in order to decrease development and testing costs and to make it easy for developers to integrate it into their own products.

## Non-root installation

In some deployments, the system administrator may require that Mt Wilson is installed as a non-root user in order to avoid granting root privileges to subordinates (people or software) who may be tasked with installing Mt Wilson. Therefore, a non-root installation must also be possible while recognizing that some actions cannot be automated and must be performed by the authorized root user prior to non-root installation.

# Architecture

## Installation Phases

### Extract Installer

The installer and all its resources are extracted into a temporary directory. This temporary directory will be deleted when the installer exits. Although files are written to disk during this step, they are of a temporary nature.

**The system is not modified during this step.**

### Check Environment

This step gathers system information and conducts a pre-install conditions check. For example, in this phase the installer determines whether it is running as a root or non-root user. **The system is not modified during this step.**

The installer also reads in the “environment file”, a file which is placed in the current user’s home directory and is used to provide non-interactive user input to the installer. Options in the environment file can alter target install location, configuration settings, selected features, and affect post-install actions. The environment file is a convenience for the administrator to enable consistent, repeatable, non-interactive installations.

The environment file is actually a shell script that defines environment variables. The “export” command is optional, and either way the definitions will only be available to the installer (not to the user running the installer, unless they source the environment file themselves). Because it’s a shell script, the variable definitions can be dynamic, for example X=5 and Y=$X would result in Y=5.

After reading the environment file, this phase may use default values for any variables that have not been set by the environment.

The installer may also check if the application has already been installed, and read any environment variables that are defined by the installed application reflecting prior configuration. The installation paths for the prior version should be used, even if they are different from the default installation paths for the new version. For example, upgrading from 3.0 (default install path /opt/mtwilson) to 4.0 (default install path /opt/install/mtwilson) will install the new 4.0 files into /opt/mtwilson because it’s already the configured path.

The environment file is optional so it is not an error if it isn’t present.

All non-password settings should have a reasonable default to avoid prompting the user for a lot of inputs when the environment file is missing. Passwords cannot have a default value so user must be prompted for any passwords they have not already set in the environment.

If a master password is required and is not in the environment, the installer must prompt for it and exit if it’s not input by the user.

### Accept License or Legal Notice

The installer must display the license for the software and subcomponents as required. An environment variable could be used to automatically accept the license for non-interactive installs, for example ACCEPT\_ALL\_LICENSE\_TERMS=yes

**The system is not modified during this step.**

### Create application user and directory

When installing as root, the installer can automatically create a non-root user and the application directory. Both non-root username and application directory are configurable via the environment file.

When installing as a non-root user, the non-root user must already be created and the application directory must already be created or it must be configured to be in a location to which the non-root user has write access.

If at the end of this phase the application directory does not exist, the installer must print an error message and exit.

This step is skipped when upgrading existing software because we reuse existing user and directory.

### Stop services (upgrade)

This phase is always executed but only applies during an upgrade. It would be skipped during initial installation.

If the application is already installed, and it includes a process monitor like Monit, then the process monitor must be informed that the monitored application services are going to be in maintenance mode. If the process monitor doesn’t have this feature, then it must be stopped, reconfigured to remove our service definitions, and then restarted. This will prevent the service monitor from attempting to restart our application during the upgrade.

Then the application services must be stopped to ensure that all locks are removed and any old files can be deleted.

### Check network port availability

The network ports required by the application services must be checked to ensure they are not in use. For an upgrade, the existing service was stopped by a prior step so nothing should be using the configured ports. If another program is bound to the ports, the installer must print an error and exit.

**The system is not modified during this step.**

### Backup configuration files (upgrade)

Configuration files from an existing installation are now backed up to a folder in the new installation directory’s repository.

### Delete obsolete application files (upgrade)

This phase is always executed but only applies during an upgrade. It would be skipped during initial installation.

Any application files from the prior installed version that are no longer needed are deleted.

This includes deleting all jar files from the application Java directory.

### Install dependencies

The software may depend on other software which may be available via the package manager. When installing as root, the installer calls the package manager to automatically download and install these dependency packages.

Other dependencies may be included with the installer because they are not be available via the package manager or because the version available via the package manager is not suitable.

The installer would install these dependencies next.

### Copy application files

All application files must be copied from the installation package to the application directory. They can then be used by subsequent phases from the application directory.

After this phase, it should be safe to unmount the installation medium or delete installation temporary files.

Copying files should be limited to “unzip”, “untar”, and “cp” commands. Executable installers should not be run during this step.

### Write application environment variables

The variables that will be needed by the application at run time should be written to a folder within the application directory. This allows the application to be aware of location of other components without having to detect at every start, and also informs future upgrades where things are located. Customizations such as placing the repository folder in a custom location, or placing the configuration folder in a custom location, can be implemented by setting those variables.

### Run automated setup tasks

This uses binaries and scripts already copied to the application folder, and any application environment variables.

If any tasks need to import configuration from the previous version (upgrade only) they can get it from the backup location.

### Post-install integrations

In this phase, the installer running as root can automatically register startup scripts, open required firewall ports, and lock down the application non-root user account. Any actions requiring root permissions would not be possible during a non-root install and it would be the responsibility of the administrator to do what is needed using the guidance in the installation guide.

### Start services

Installed software that should run as a service is started at the end of the installation.

## Distribution

### Self-extracting installer

Mt Wilson is comprised of numerous Java archives, shell scripts, HTML content, and executable binaries. These are packaged into a zip file. This zip file is wrapped with an installation script and a self-extracting archive.

### Linux Package

Mt Wilson may also be available as a Linux package in Ubuntu, RedHat, SUSE, etc. Each Java archive, executable binary, and shell script collection might be packaged separately to let the package manager automatically install them as dependencies of the main package.

### Windows Package

Mt Wilson may also be available as a Windows Installer (.msi) package.

## Un-installation

A regular uninstall should:

* Delete application code and binaries
* Remove startup scripts
* Remove any integration with other system components

A “purge” uninstall should perform above steps and also:

* Delete configuration files
* Delete logs
* Delete user data maintained by the application (database or repository folder)

It is the user’s responsibility to export and backup their configuration, data, and logs before they uninstall with the purge option.

# Development Roadmap

Currently the “functions.sh” file contains functions to assist in the installation and setup of many features. This needs to be separated into one shell script per feature. For better performance, these separate scripts can be combined into a single functions.sh every time they are updated.

## Windows

On the build server, we can install WiX, Mono, and Wine together to allow us to integrate Windows Installer (.msi) generation into the build process.

WiX <http://wixtoolset.org/> license “Microsoft Reciprocal License (MS-RL)” for the tool which allows us to use and copy the WiX tool for free, and makes no claims on the output (our installer). <http://wixtoolset.org/about/license/>

Mono <http://www.mono-project.com/> is used while running WiX only and would not be distributed as part of our application. License <http://www.mono-project.com/docs/faq/licensing/>

Wine <https://www.winehq.org/> is used while running WiX only and would not be distributed as part of our application. License <https://www.winehq.org/license>