Mt Wilson

Configuration

# Background

Mt Wilson is both an enterprise product and a datacenter product. Customers require varying degrees of automation, flexibility, and security for its configuration. This blueprint generally describes how to configure Mt Wilson. Each feature should follow this blueprint and document its own specific requirements.

# Architecture

The environment provided by the operating system is used to bootstrap the configuration and determine the home directory, configuration directory, and features directory. Without environment settings, the application uses built-in defaults.

Using the folder locations determined by the environment or defaults, the application configuration is loaded (and decrypted if necessary) and is available to all components via a static configuration factory. The application configuration is limited to key-value pairs.

Any other type of configuration, such as a keystore file or other resource, is by its nature non-generic and belongs to a specific feature. Each feature can load its specific configuration files either from the application configuration folder or from its own feature folder. Use of the application configuration folder must be kept to a minimum.

The mtwilson-util-configuration project defines a general API for Configuration objects and related utility classes that is the preferred method for defining and sharing configuration information throughout Mt Wilson.

## Environment Variables

All use of environment variables can be found by looking for usage of java.lang.System.getenv and com.intel.mtwilson.Environment.get.

The System.getenv function can be used to retrieve all environment variables.

The Environment.get functions are used to retrieve application-specific environment variables. These are defined by a prefix. The default prefix is “MTWILSON\_” and an example variable is “MTWILSON\_HOME” which can be obtained by calling Environment.get(“HOME”). This use is encouraged because it abstracts the application name and allows the same feature to work consistently across different applications such as Mt Wilson attestation server, Key Server, Key Server Proxy, Trust Director, and Trust Agent. These applications can then define application-specific variables in accordance with industry conventions, run simultaneously on the same server, share the same launch and configuration code, yet be configured separately.

## Configuration File

The configuration file may be encrypted. It will be loaded (and decrypted if necessary) by the launcher.

# Configuration API

The Configuration interface contains the following methods:

keys() returns Set<String> of all available configuration keys

get(String key) returns String with value of the requested key or null if the key is not defined

get(String key, String defaultValue) returns String with value of the requested key or defaultValue if the key is not defined

set(String key, String value) sets the value for the requested key and does not return anything

isEditable() returns boolean with true if set(key,value) should work and false if set(key,value) will throw an UnsupportedOperationException

The mtwilson-util-configuration project contains some useful decorators such as a ReadonlyConfiguration, a ValveConfiguration, a LayeredConfiguration, and a CommonsConfigurationDecorator (which provides a commons-configuration front-end to the Configuration API).

## How to get the configuration

## Mt Wilson

In Mt Wilson, the application configuration is available via a static function My.configuration() and is provided in a read-only view. Setup tasks or commands that need to edit the configuration can load it using the ConfigurationProvider, make changes, and then store it using the ConfigurationProvider. If this is done within the running application, the static configuration should be reloaded by calling My.reset(). All subsequent reads of My.configuration() will yield the new configuration… but because some components may cache configuration settings, the only way to ensure that all components are using new settings is by restarting the application.