

Variables

Variables - `<variables>`

Understanding Variables

This element allows you to define variables for the variables substitution system. Some variables are built-in, such as `$INSTALL_PATH` (which is the installation path chosen by the user).

To define a set of variables, you place as many `<variable>` elements as needed inside a `<variables>` or `<dynamicvariables>` element which are in turn children of the `<installation>` element.

If you define a variable named `VERSION` you need to type `$VERSION` in the files to parse. The variable substitutor will then replace it with the correct value.

Each `<variable>` tag take the following attributes :

- **name** : the variable name
- **value** : the variable value

Here's a sample `<variables>` section :

```
<variables>
  <variable name="app-version" value="1.4"/>
  <variable name="released-on" value="08/03/2002"/>
</variables>
```

Types of Variables

Static variables

Static variables can be defined by the user using the `<variables>` element.

They are evaluated on launching an installation and won't alter during the installation process. Static variables can be assigned from a system environment variable on launching an IzPack installations.

Environment variables

Environment variables can be accessed using the syntax `#{ENV[variable]}`. The curly braces are mandatory. Note that variable names are case-sensitive and usually in UPPER CASE. For example, to get the value of the OS environment variable `ANT_HOME`, use `{ENV[ANT_HOME]}` in the resource you want to substitute the according value during an installation.

Dynamic variables

Dynamic variables can be defined by the user using the `<dynamicvariables>` element in the installation description.

Dynamic variables are the most powerful facility of saving and gathering values on a target system where the installation is launched. The value of dynamic variables will be evaluated every time a panel is switched.

Furthermore, beginning with IzPack 5.0, dynamic variables can be also assigned from several configuration and

archive files, the Windows registry, the output of a command execution and dynamically filtered using Java regular expressions. See [Dynamic Variables](#) for more information and some examples how this can be achieved. Dynamic variables can be handled like other variable types for variable substitution.

Built-In variables

There is a couple of "hard-coded" variables implemented in IzPack. The value of several built-in variables might alter depending on local conditions on the target system where the installation is launched.

The following variables are built-in :

- **\$INSTALL_PATH**: the installation path on the target system, as chosen by the user
- **\$INSTALL_DRIVE**: the drive letter part of the installation path on the target system, set on Windows systems, only
- **\$APPLICATIONS_DEFAULT_ROOT**: the default path for applications
- **\$JAVA_HOME**: the Java™ virtual machine home path
- **\$CLASS_PATH**: the Class Path used mainly for Java Applications
- **\$USER_HOME**: the user's home directory path
- **\$USER_NAME**: the user name
- **\$APP_NAME**: the application name
- **\$APP_URL**: the application URL
- **\$APP_VER**: the application version
- **\$ISO2_LANG**: the ISO2 language code of the selected langpack.
- **\$ISO3_LANG**: the ISO3 language code of the selected langpack.
- **\$IP_ADDRESS**: the IP Address of the local machine.
- **\$HOST_NAME**: the HostName of the local machine.
- **\$FILE_SEPARATOR**: the file separator on the installation system
- **\$DesktopShortcutCheckboxEnabled**: When set to true, it automatically checks the "Create Desktop Shortcuts" button. To see how to use it, go to `The Variables Element <variables> Be careful this variable is case sensitive !
- **\$InstallerFrame.logfilePath**: The path to the install log. This file contains the paths of all installed files. If set to "default" then the "\$INSTALL_PATH/Uninstaller/install.log" path will be used. To see how to use it, go to `The Variables Element <variables>. If this variable is not set, no install.log will be created.
- **\$TargetPanel.dir.<platform>**
For setting <platform>, see also: [Use Cases](#).
Defines the fully qualified target installation directory per target platform. IzPack choses the most matching one according to the current platform from the list during an installation.

Variable Substitution

References to variables enclosed in certain placeholder begin and end marks (for example `{` and `}`) can be used to substitute the according placeholders in

- attribute values and embedded text in the installation description from which an IzPack setup is created
- resource files of an installation
- installed text files and shell scripts in several formats using the <parsable> tag

Substitute Variables in the Installer Descriptor

Additionally to [properties](#), static variables defined by the <variables> tag can be used to be substituted in the installer descriptor (install.xml) itself, where they have been defined.

This is limited to static and built-in variables. It is not possible to substitute [dynamic variables](#) here, because they are refreshed and evaluated during the installation as soon as a panel is activated or changed, not at compilation time.

Variable references in the installer descriptor apply on the plain style syntax, with a leading \$ before the variable name, optionally enclosed in curly braces. Example `${MY_VAR}` or `$MY_VAR`.

Substitute Variables in Resource Files

Variables can be also substituted in resource files declared by the `<resources>` tag.

This is limited to static and built-in variables. It is not possible to substitute [dynamic variables](#) here, because they are refreshed and evaluated during the installation as soon as a panel is activated or changed, not at compilation time, and resource files are static files.

Variable references in the installer descriptor apply on the plain style syntax, with a leading \$ before the variable name, optionally enclosed in curly braces. Example `${MY_VAR}` or `$MY_VAR`.

Substitute Variables in Installed Files

For replacing variable references in installed textfiles by their current variable values those files must be tagged as *parsable*.

For this purpose, apply the `<parsable>` tag on a file previously included in the installation. Files marked *parsable* are parsed during the installation on the fly, references to existing variables are substituted, and the file is saved with substituted variable values.

See the [<packs> element documentation](#) for more information on how to use the `<parsable>` tag.

Using System Properties As Variables

System properties given on the command line can be directly included as IzPack variables when using the `SYSTEM_` prefix followed by the system property name in the variable name, and substituting all occurrences of `'.'` by `'_'`. System properties with null values are not substituted.

Examples:

- `$SYSTEM_java_version`
- `$SYSTEM_os_name`