Variables

Variables - `<variables>`

Understanding Variables

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

To define variables, you place **`<variable>`** elements 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 value that you specified in the **`<variables>`** or **`<dynamicvariables>`** element with the name VERSION.

Each **`<variable>`** tag takes the following attributes :

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

Note that variable names are case-sensitive.

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 are defined using the **`<variables>`** element.

Static variables are evaluated when an installation starts and will not alter during the installation process.

Environment variables

System environment variables that are defined when the IzPack installation is launched, can be used without a **`<variables>`** element definition **.**
Environment variables can be accessed directly using the syntax **`${ENV[variable]}`**. The curly braces are mandatory.
Environment variable names are usually in UPPER CASE. For example, to get the value of the OS environment variable ANT_HOME, use ${ENV[ANT_HOME]} to substitute the definition of ANT_HOME.

Dynamic variables

The value of dynamic variables will be evaluated every time a panel is switched. This makes them useful for passing information between panels during the installation.

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

Dynamic variables are handled like other variable types for variable substitution.

See Dynamic Variables for more information and some examples how this can be achieved.

Built-In variables

There are a couple of "hard-coded" variables implemented in IzPack.

The following variables are built-in and have values set by IzPack depending on the target system or information supplied by the user during the installation.

- **`$INSTALL_PATH`**: the installation path on the target system, as chosen by the user
- **`$INSTALL_DRIVE`**: the drive letter part of the installation path on the target system, applies to Windows systems only.
- **`$APPLICATIONS_DEFAULT_ROOT`**: the default path for applications
- **`$JAVA_HOME`**: the JavaTM 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.
- **`$FILE_SEPARATOR`**: the file separator on the installation system
- **$TargetPanel.dir.*<platform>***
  For setting <*`platform`*>, see also: `Use Cases`.
  the fully qualified target installation directory for the target platform. IzPack choses the most closely matching directory according to the current platform..

The following variables are defined but can have values redefined by the installation procedure:

- **`$DesktopShortcutCheckboxEnabled`**: When set to true, it automatically checks the "Create Desktop Shortcuts" button. To see how to use it, go to `The Variables Element <variables> Be careful this variable is case sensitve !
- **`$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.

Variable Substitution

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

In additional to properties, static variables defined by the **`<variables>`** tag can 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

To replace variable references in installed textfiles those files must be tagged as *parsable*.

For this purpose, specify the **`<parsable>`** tag on a file. 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 are directly assigned to variables using the following syntax, provided the system property is `variable.name`:

${SYSTEM[*`variable.name`*]}

Example:

If `"-Dfeature.Enabled=true"` is on the command line that launched the installer, the command line argument can be used as follows:

Example 1

```
<variables>
  <variable name="featureEnabled" value="${SYSTEM[feature.Enabled]}" />
</variables>

<conditions>
  <condition type="variable" id="isFeatureEnabled">
    <name>featureEnabled</name>
    <value>true</value>
  </condition>
</conditions>
```

is equivalent to:

Example 2

```
<conditions>
  <condition type="variable" id="isFeatureEnabled">
    <name>SYSTEM[feature.Enabled]</name>
    <value>true</value>
  </condition>
</conditions>
```

The format used in IzPack 4 and earlier is supported in IzPack 5.0 for backward compatibility:

${SYSTEM_*`variable_name`*}

If the variable name contains '.' characters they got to be replaced by '_' here.

Examples

If `"-Dfeature.Enabled=true"` is on the command line that launched the installer, the command line argument can be used as follows:

Example 1

```
<variables>
  <variable name="featureEnabled" value="${SYSTEM_feature_Enabled}" />
</variables>

<conditions>
  <condition type="variable" id="isFeatureEnabled">
    <name>featureEnabled</name>
    <value>true</value>
  </condition>
</conditions>
```

is equivalent to:

Example 2

```
<conditions>
  <condition type="variable" id="isFeatureEnabled">
    <name>SYSTEM_feature_Enabled</name>
    <value>true</value>
  </condition>
</conditions>
```