Variables Overview

Overview

Variables in Automise are the key to making your builds dynamic. Variables can be used in almost every text property of every Automise action, for example in fields that specify file paths, directories etc.

There are four types of variables available:

- Local variables are defined on Action Groups, and only available to child actions.
- Project variables are defined by the script writer for a single project.
- Application variables are defined by Automise. These give useful information about the context of the project and installed tools.
- Environment variables provide access to Windows environment variables such as PATH and OS. The range of environment variables available depends on the installation of Windows and other software

Variables can be modified at run time by using actions, or script.

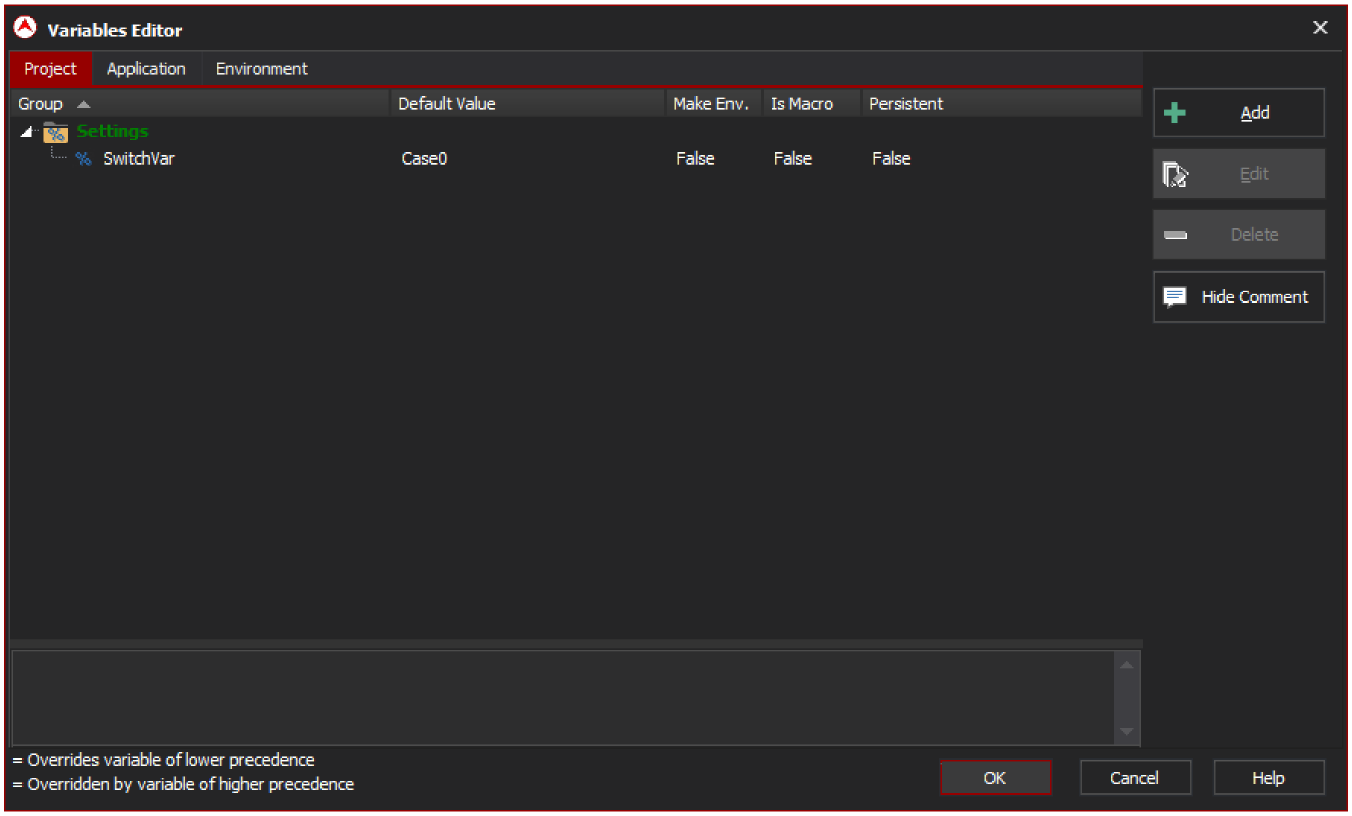
Variables defined in Automise can be referenced in Automise VBScript and JavaScript in the same way as normal script variables. In PowerShell, variables can be referenced by using the following syntax:

```
$FBVariables.GetVariable(<variablename>)
```

```
$FBVariables.SetVariable(<variablename>)
```

Variables Editor

To create or edit existing variables, use the Variables Editor. You can launch the Variables Editor from the Project menu by selecting Edit Variables, with the keyboard short cut <Shift+F2>, or by double clicking on the Variables node of the Project Tree.



Group	Default Value	Make Env.	Is Macro	Persistent
Settings				
SwitchVar	Case0	False	False	False

= Overrides variable of lower precedence
= Overridden by variable of higher precedence

In addition to the three categories of variables, there are three flags that can be set on each variable.

Make Env.

Project and User Variables can be made available as environment variables to applications that are executed by Automise. That is, if you create a variable MYVAR and set the "Make Env." flag, a command shell action could access it as %MYVAR%.

Is Macro

The Is Macro flag forces Automise to re-evaluate the variable by expanding the Default Value, whenever that variable is referenced during the build. This is particularly useful for variables that are used in scripting: whereas most actions automatically expand all variable references, regardless of the "Is Macro" flag, this does not happen in script events. If you set this flag, however, any contained variable references are automatically expanded.

For example, you can make a "Buildpath" macro variable that contains "%BuildHome%\Automise\%BuildName%". Each time it is accessed, whether from script or an action, it will expand to the current value of those variables.

Macro Variables cannot be set during the build using the Set Variable action or any other means.

Persistent

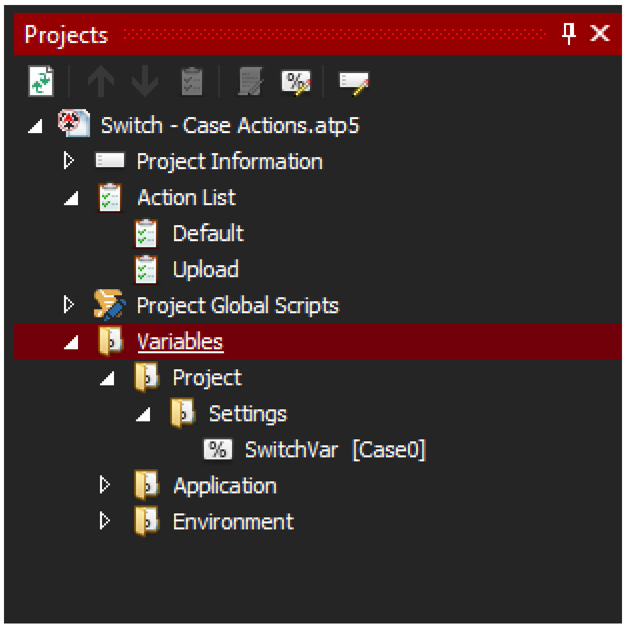
The values of persistent variables are stored automatically between executions of the Automise project. Persistent variables are saved regardless of whether or not the project is saved.

Persistent variable values are stored in .fbpinf files. Each fbpinf file has the same name as the Automise project (to persist variables, Automise needs write access to the directory containing the project and the ability to create or write to the .fbpinf file.)

Groups

Variables can be organised into variable groups to assist with managing your project. The grouping has no effect on variable scope or behaviour at runtime.

In the following example, variable "ObjName" is in the group "BuildVars.Debug".



See Also

Using Variables

Project Variables

Environment Variables

Application Variables

Action List Parameters

Escaping Variable References