RUDDER

docs.rudder.io

Conditions

Conditions are context entities represented by a string, that can be either set or not set, depending on context. They allow using conditions to generic method usage.
Variable name must match

```
[a-zA-Z0-9][a-zA-Z0-9_]*
```

Available conditions

Group conditions: defined only if the node is in the given group (available in the group details)

```
group_group_uuid
group_group_name
```

System conditions: various system information defined by default

```
centos_7, ubuntu_14_04
```

Result conditions: defined by the execution of another generic method (available at the bottom of the generic method call configuration)

```
generic_method_name_parameter_value_kept
generic_method_name_parameter_value_repaired
generic_method_name_parameter_value_error
```

Conditions manually defined in the agent call

```
rudder agent run -D my_condition
```

Group

```
(condition_expression)
```

And

```
condition.other
```

Or

```
condition|other
```

Not

```
!condition
```

Paths

On the nodes

Policy server configuration file

```
/var/rudder/cfengine-community/policy_server.dat
```

On the server

Directory containing all the configuration policies in a git repository

```
/var/rudder/configuration-directory/
```

Directory shared to Nodes from the Server

```
/var/rudder/configuration-directory/shared-files/
```

Directory containing the configuration events (changes and errors)

```
/var/log/rudder/compliance/non-compliant-reports
```

Commands

To update the policies and enforce them

```
rudder agent run -u
```

To see detailed output

```
rudder agent run -i
```

To trigger an inventory

```
rudder agent inventory
```

Other commands and options

```
man rudder
```

Variables

Variable name must match

```
[a-zA-Z0-9][a-zA-Z0-9_]*
```

Variables in Directives parameters are evauated at generation on the server, exceptions are tagged with execution
Variables in the Technique Editor are evaluated at execution on the nodes
Node properties can be overriden at execution on the nodes using files containing a "properties" object placed in

```
/var/rudder/local/properties.d/*.json
```

Only in Directives

System variables about a node

```
${rudder.node.id}
${rudder.node.hostname}
${rudder.node.admin}
```

System variables about a node's policy server

```
${rudder.node.policyserver.id}
${rudder.node.policyserver.hostname}
${rudder.node.policyserver.admin}
```

Node properties

```
${node.properties[key]}
${node.properties[subtree]}
${node.properties[key] | node } execution
```

Default values (only with node properties)

```
${variable | default = "value" }
${variable | default="value"|default="fallback" }
${variable | default = """value with "quotes" """ }
${variable | default = ${any_other_variable} }
```

Javascript Engine (with any variable)

```
"${variable}".substring(0,3)
```

Rudder Javascript library

```
rudder.hash.md5/sha256/sha512(string)
rudder.password.auto/unix/aix("MD5/SHA256/SHA512",
password [, salt])
```

In Directives and in the Technique Editor

Global Parameters

```
${rudder_parameter.string_name}
```

From the "Variable (string)" technique

```
${generic_variable_definition.string_name}
```

From the "Variable from command output (string)" technique

```
${generic_cmd_var_def.string_name}
```

From the "Variable from JSON file (dict)" technique

```
${variable_prefix.dict_name[key]}
```

Node properties

```
${node.properties[key]}
${node.local_custom_properties[key]}
```

Only in the Technique Editor

User Variables defined using generic methods

```
${variable_prefix.string_name}
${variable_prefix.iterator_name}
${variable_prefix.dict_name[key]}
```

Mustache Templating

Conditions

(no condition expression here)

```
{{#classes.condition}}
condition is defined
{{/classes.condition}}
```

```
{{^classes.condition}}
condition is not defined
{{/classes.condition}}
```

Variables

```
{{{vars.node.properties.variable_name}}}
{{{vars.generic_variable_definition.variable_name}}}
{{{vars.variable_prefix.string_name}}}
{{{vars.variable_prefix.dict_name.key}}}
```

Iteration

```
{{#vars.variable_prefix.iterator_name}}
{{{.}}} is the current iterator_name value
{{/vars.variable_prefix.iterator_name}}

{{#vars.variable_prefix.dict_name}}
{{{@}}} is the current dict_name key
{{{.}}} is the current dict_name value
{{/vars.variable_prefix.dict_name}}

{{#vars.variable_prefix.dict_name}}
{{{.name}}} is the current dict_name[name]
{{/vars.variable_prefix.dict_name}}
```

Jinja2 Templating

Conditions

(no condition expression here)

```
{% if classes.condition is defined %}
condition is defined
{% endif %}
{% if not classes.condition is defined %}
condition is not defined
{% endif %}
```

Variables

```
{{ vars.variable_prefix.my_variable }}
```

Iteration

```
{% for item in vars.variable_prefix.dict %}
{{ item }} is the current item value
{{ item.key }} is the the current item[key] value
{% endfor %}

{% for key,value in vars.prefix.dict %}
{{ key }} has value {{ value }}
{% endfor %}
```