RUDDER

https://www.rudder-project.org/doc

Classes

Classes 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 classes

Group classes: defined only if the node is in the given group (available in the group details)
`group_group_uuid`
`group_group_name`
System classes: various system information defined by default
`centos_7`, `ubuntu_14_04`
Result classes: 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`
Classes manually defined in the agent call
`rudder agent run -D my_class`

Classes operators

Group
`(my_class_expression)`
And
`my_class.my_other_class`
Or
`my_class|my_other_class`
Not
`!my_class`

Paths

On the nodes

Directory containing the agent run logs
`/var/rudder/cfengine-community/outputs`
Last run log
`/var/rudder/cfengine-community/outputs/previous`

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

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}`
Global parameters
`${rudder.param.parameter_name}`
Node properties
`${node.properties[key]}`
`${node.properties[subtree]}`
`${node.properties[key] | node }` execution
Default values (only with node properties)
`${variable | default = "value" }`
`${variable | default = """value 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

Classes

(no class expression here)

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

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

Variables

```
{{{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

Classes

(no class expression here)

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