Device Variables

Unimus supports declaring per-device variables, which can be used in Config Push for more automation flexibility.

Please note variables are available starting in Unimus 2.2.0.

Defining device variables

Device variables can be defined for device(s) in the "Devices" screen. You will find a "Variables" button here, which opens the variable edit window.

The Variable edit window has 2 separate modes:

- single device variable editing mode
- bulk (multi-device) variable editing mode

If you open the Variables window WITHOUT any device being currently selected, you will enter into the bulk edit mode. If you open the window when multiple devices are selected, you will enter the bulk edit mode with data pre-filled for devices which were selected. Finally, if a single device is selected, you will enter into single device variable edit.

Single device variable edit

In this mode, you can set variables for the specific device which was selected when you opened the variable edit window. You can simply specify variables in a "key=value" format. If any variables are already present on the device, they will be loaded into the edit window for you. Here is an example of how variable configuration on a single device would look like:

```
foo = bar
variable_name = variable_value
hostname = router
```

The above configuration would set 3 variables for the selected device, "foo", "variable_name" and "hostname". When saving variables, omitting any existing variables removes those variables. To simplify - when you save variables for a device, all existing variables will be removed, and then variables will be set to the device as provided in the edit window.

Bulk (multi-device) variable edit

In bulk mode, you need to specify for which device you wish to set variables. Each device is specified as a section, and after opening a section for a specific device, you can set its variables. You can then move on to another device by opening another section. Here is an example:

```
[device_address_1 @ Z1]
variable_name = variable_value
foo = bar

[192.168.1.1 @ Z2]
hostname = router
```

In the example above, we set variables for 2 devices:

- device "device_address_1" in a Zone with ZoneID "Z1"
- device "192.168.1.1" in a Zone ZoneID "Z2"

As mentioned in a previous section, if you open the variable edit window without any devices selected, it will be in an empty bulk-edit mode. You don't need to manually write out the device sections.You can select multiple devices in the Device table, and open the variable edit window to have the devices pre-filled for you, including their existing variables.

Please note that when saving devices in bulk edit mode, omitting variables for devices which already have variables present will remove those variables. To simplify - if you save variables for devices in bulk edit mode, all existing variables from each specified device will be removed, and

then variables will be set to the devices as provided from the new configuration.

Variable formatting

The variable name, as well as the variable value will have both their leading and tailing white-space removed. All of these declarations as valid, and equal:

```
hostname=router1
hostname = router1
hostname= router1
```

All of the above will result in a variable with name "hostname" and value "router1" created. It is not possible to declare variables which contain leading or tailing spaces. Spaces inside both variable names and values are however fully supported.

Using variables in Config Push

Variables can be used in Config Push Commands to substitute unique per-device values when commands are being pushed to devices. This allows you to create a single Push Preset which can send unique commands to each device, or allow for unique per-device values to be inserted inside commands. Variables can be referred to in Config Push in the following format: "${variable_name}".

Here is an example which would utilize a variable with name "snmp_community" to create an SNMPv2 community on MikroTik RouterOS:

```
/snmp community
add security=none write-access=no name=${snmp_community}
```

Alternatively, here is an example of creating a user on Cisco IOS, where the password for the user is read from a device variable:

```
! simple user creation
username network-admin password ${admin-password}

! privileged user creation
username network-admin privilege 15 password ${admin-password}
```

Variable substitution and the "\" escape character

If you wish to send a sequence beginning with "${" to the device, you can escape the variable declaration like this:

```
/interface bridge
add name=lo1 comment="This is a comment with \${some strange characters}"
```

In the example above, Unimus would not substitute the text above for a variable's value. The comment for the loopback interface would be set to "This is a comment with ${some strange characters}".

Variable chaining and Push Behavior Modifiers

Variables chaining is NOT supported. In other words, you can NOT refer to other variables from within different variables. Config Push Behavior Modifiers however CAN be declared inside variables.

```
# this is NOT possible
var1=value
var2=some_other_${var1}
```