# Command parsing on Cisco IOS finding interfaces and ports

The goal of this example is to find interfaces that belong to specific vlans.

Create a new Command Job and save it using a desired name.

## Command Job

The job is “empty”. It still requires something though, so it's filled with a '!'.

```
!
```

## Parsing template parsevlan

The command that is used is *show vlan | i user|data*.

This parse template will match specifically on the word “active”. For those lines it will set the vlannumber and vlandescr variables.

```
<vlannumber> <vlandescr> active |*|
```

## Parsing template intfnumber

The command *show int switchport | i Name:|Access Mode VLAN* is used to find the combination of information of interface and (data) vlan.

In this parsing template and % sign is used to create a hash to store the other variables in relation to it. This allows to retrieve the <accessvlan> variable using the following inside a loop in the scenario: `<<interface>.accessvlan%intf>`

```
Name: <%interface>
Access Mode VLAN: <accessvlan>
```

Make sure to read the hash article on how <variable> is different than <%variable>.

## Job scenario

```
Description Parsing vlan and interfaces on <node>
task := scn_parse_vlan_intf
```

## Scenario scn_parse_vlan_intf

```
# Test whether node is live and reachable
```

```
reachable -n <node>
if <error>
    LogAction -n <node> -a Parse_job -m "<node> is not reachable"
    stop
endif
LogAction -n <node> -a Parse_job -m "<node> is reachable"


<%cmd> := parse_cmd -n <node> -t parsevlan -r "show vlan"
<vlans> := keys <%cmd>

foreach <vlan> in <vlans>
    log -m "vlan: <vlan>"
    <%intf> := parse_cmd -n <node> -t intfnumber -r "show int switchport \\|
i Name:\\|Access Mode VLAN"

    <interfaces> := keys <%intf>
    foreach <interface> in <interfaces>
        log -m "interface: <interface>"
        log -m "vlan: <<interface>.accessvlan%intf>"
        if "<<interface>.accessvlan%intf> == <vlan>"
            log -m "Interface found, generating and/or appending
configuration"
            config_create -n <node> -t user_interface -f
<node>_user_intf.cmd -x
            if <error>
                Logaction -n <node> -m "Failed to create the template for
<node>"
                Stop
            endif
    endeach
endeach

log -m "Pushing new configuration"
config_exec -n <node> -f <node>_user_intf.cmd
if <error>
    Logaction -n <node> -m "Failed to configure <node> commands"
    Stop
endif

end
```

Some things to note:

- `<interfaces> := keys <%intf>`, this is required to be able to loop over the created hash/dictionary.
- Config is being generated for every interface in addition and once completely done, it will be pushed to the device.
- `|` (pipes) need to be escaped (double). This is because the can also be used as conditionals.