ASE Cheat Sheet (last modified 12/6/16)

For the latest version go to: https://ase.arubanetworks.com/docs

Field or Variable

- Creates a variable/field
- Sets a field input for the user at this location in the form
- Will print or output field in the config

```
(%variable%)
```

Field or Variable Related Tags

(#nonconfigvar#)

- Creates a variable/field
- Set variable input at this location
- Does not print or output the variable in the config

```
(# nonconfigvar variable #)
```

- Options:
  - variable
    - name of variable; required

(#defaults#)

Option to auto populate form with defaults using the Actions dropdown; conventionally is the first line in configuration text, if used

```
(# defaults "EXAMPLE VALUES" variable1="VALUE1"
variable2="VALUE2" #)
```

- Options:
  - "EXAMPLE VALUES"
    - name of the set; required
  - variable1="VALUE1"
    - variable1 is likely an already existing variable in the solution; additional variables added as needed

Grouping Tags

(#begingroup#) (#endgroup#)

Input field groupings appearing in single form. Each begingroup appears as sequential tabbed sections.

```
(# begingroup title="TAB LABEL" order="10" #)
CONTENT (%variable1%)
(# endgroup #)
```

- Options:
  - title="TAB LABEL"
    - generally required
  - order="10"
    - optional user-defined number; arbitrary ordering defined by user
  - description="Create SSID and LMS IP and ACL"
    - optional longer description displayed to user when in form

(#begindevice#) (#enddevice#)

Output config groupings. Each begindevice are configs sets applied to different devices.

```
(# begindevice name="DEVICE1" #)
CONTENT
(# enddevice #)
```

- Options:
  - name="DEVICE1"
    - generally required

Control Structure "If" Tags

(#beginif#) (#endif#)

Enclosed block IF statement

```
(# beginif condition="field_variable.value=='STRING'" #)
CONTENT
(# endif #)
```

(#optionalline#)

One line IF statement

```
OUTPUT LINE (# optionalline
condition="field_variable.value==true" #)
```

- Options applying to both tags above. Two examples:
  - condition="field_variable.value==true"
  - condition="field_variable.value=='STRING'"

Control Structure "Loop"-like Tag

(#hook#)

Generates a block of config, repeated for each hook_to variable. The hook_to variable is generally an already entered field/variable containing multiple values used in the hook. Javascript is required to parse this value and resulting config is generated and repeated for each multiple value in the hook_to variable.

```
(# hook HOOK_NAME hook_to="INPUT_FIELD" #)
# Implementation in Javascript is in a textbox in the hooks
section
```

This is similar to invoking a function:

- HOOK_NAME(INPUT_FIELD)
  - config commands INPUT_FIELD[1]
  - config commands INPUT_FIELD[2]
  - ...

```
(# hook process_dhcp_exclude_ranges hook_to="dhcp_exclude_ranges"
template_str="ip dhcp excluded-address {0} {1}\n" #)
```

- Options with examples:
  - <hookname>
    - required
  - hook_to="dhcp_exclude_ranges"
    - required
  - template_str="ip dhcp excluded-address {0} {1}\n"
    - optional formatting string; often of a specific javascript return value format

Idioms

```
(# nonconfigvar checkbox_var #)
CONFIG (# optionalline condition="field_checkbox_var.value == true" #)
```

Checking the optional field checkbox will apply additional config

```
(# nonconfigvar optional_var #)
(# beginif condition="field_optional_var.value!=''" #)
CONFIG
(# endif #)
```

Adds additional configuration if optional variable is entered in fields