SmartCS

SmartCS x Ansible Linking

Explanatory material

About This Material

SEIKO

This material explains about SmartCS modules for Ansible and the linking with network device vendor modules, which creates a link between the SmartCS console server and Ansible.

◆Content

- Overview of SmartCS
- How to link SmartCS and Ansible
  - SmartCS modules for Ansible
  - Linking with network device vendor modules
- Related material

ANSIBLE

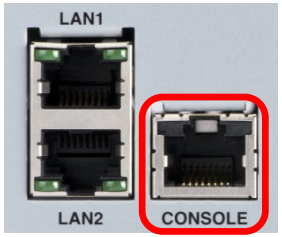
Role of SmartCS

- What is a console port?
- About the role of SmartCS

■ Role of the console port

An interface for operation using serial communications rather than IP communications

- **Initial settings**
  IP setting, user creation, SSH activation, and other initial settings
- **Operation in an emergency**
  Last access method when you are unable to access the device by IP due to the impact of a LAN port failure, network failure, etc.



RJ45

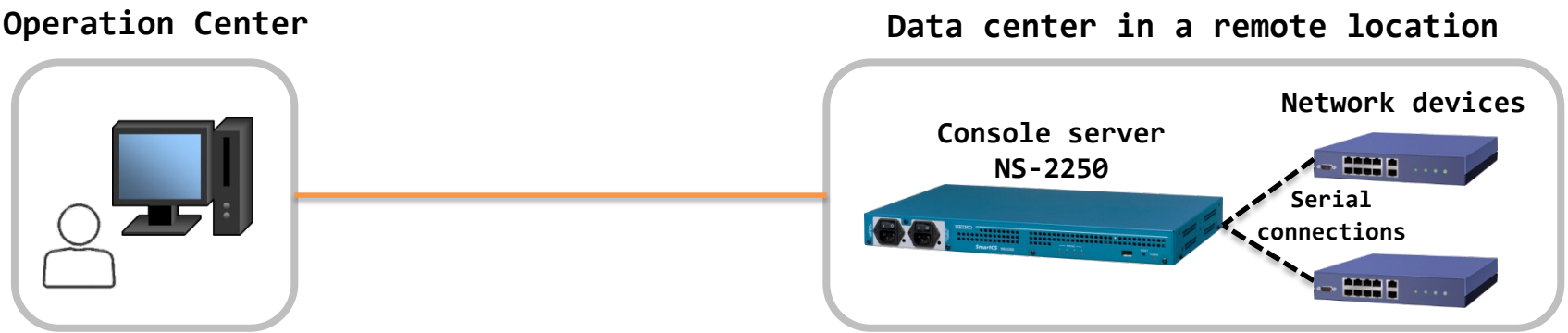
DB9

SEIKO

■ Role of SmartCS

Device which aggregates the console ports and enables remote access

- **Remote access**
  Enables remote access to a device which cannot be accessed by IP
- **Expanding the scope of operation**
  Work which is difficult to carry out remotely can be safely executed.
  ACL, routing and other setting changes, and firmware updates, etc.



Linking SmartCS and Ansible

- Required environment
- What you can do with linking
- Linking method
  - SmartCS modules for Ansible
  - Linking with network device vendor modules

ANSIBLE

Required environment

■ Required environment

Configuration

Control node
- Host OS in which Ansible is installed

Ansible
↑
"SmartCS modules for Ansible"

SmartCS
NS-2250 series
- NS-2250-16/16D
- NS-2250-32/32D
- NS-2250-48/48D

Network devices

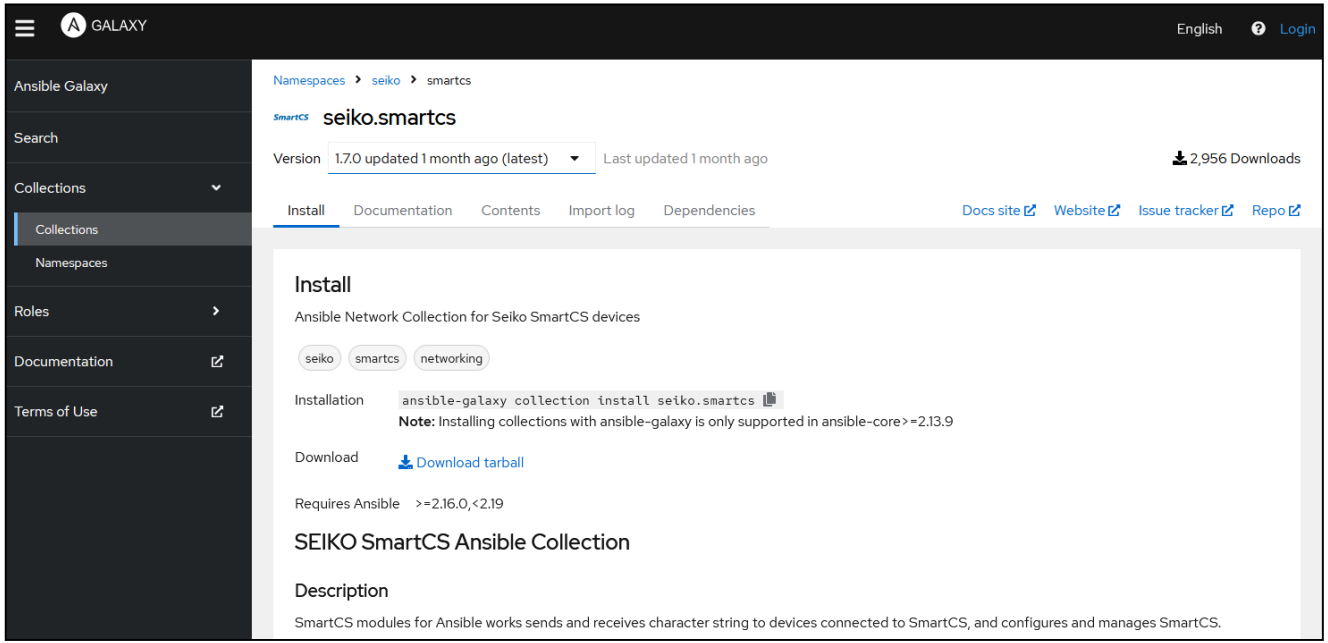
*Device which can be connected with the NS-2250

Required environment

SEIKO

■ Provision of "SmartCS Modules for Ansible"

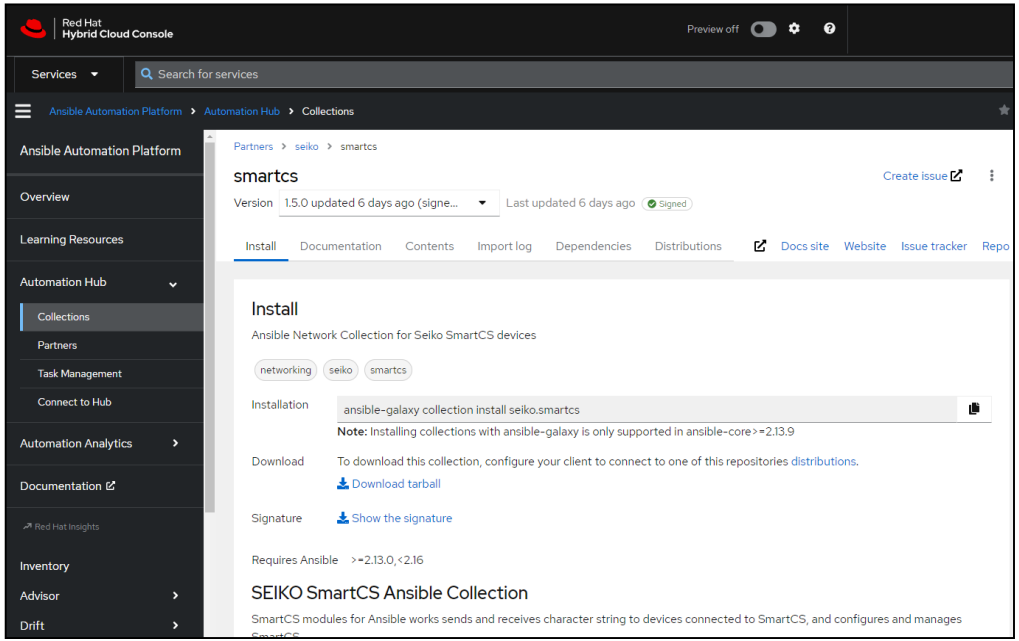
(1) Available to download and install from the Ansible Galaxy site.
https://galaxy.ansible.com/ui/repo/published/seiko/smartcs/



Required environment

■ Provision of "SmartCS Modules for Ansible"

(2) Available to download and install from the Ansible Automation Hub.
https://console.redhat.com/ansible/automation-hub/repo/published/seiko/smartcs



Required environment

SEIKO

■"SmartCS Modules for Ansible" operating environment

SmartCS modules for Ansible		Control node environment	Managed nodes environment SmartCS system software ver.	
Release	Version	ansible	NS-2250 series	NS-2240 series
2025.1	v1.7.0	ansible-core 2.16.x ansible-core 2.17.x ansible-core 2.18.x (>=2.16, < 2.19)	V2.1 and above	Not supported

*The NS-2250 software and the Ansible modules run in the combination which supports each version.

*Modules v1.3.0 and above are provided as the Ansible collections packeges.

*Modules v1.3.0 and above can be obtained from Ansible Galaxy(https://galaxy.ansible.com/seiko/smartcs).
v1.4.0 and above can be obtained from Ansible Automation Hub(https://console.redhat.com/ansible/automation-hub/repo/published/seiko/smartcs).

*Module information other than the latest version can be got from the above two site or “NS-2250 Ansible Operation Guide”.

What you can do with linking

SEIKO

■What you can do with linking

The following functions can be used by linking SmartCS and Ansible.

Function name	Module	Description
CLI Command Function	smartcs_command	Executes the SmartCS CLI commands via Ansible.
	smartcs_config	
	smartcs_facts	
Console Access Function	smartcs_tty_command	Sends characters to the console of the devices(network devices) connected to the SmartCS serial port. *This function uses the SmartCS tty manage function.
	network device vendor modules	This function links with network device vendor modules and sends characters to the console of the devices(network devices) connected to the SmartCS serial port. *This function uses the SmartCS SSH transparent connection (sshxpt function).

What you can do with linking

SEIKO

■ Scope of operation for each function

- smartcs_command
- smartcs_config
- smartcs_facts

CLI Command Function:

Main theme of this material

- smartcs_tty_command
- **Linking with** network device vendor modules

Console Access Function:

Linking method

SEIKO

■ Two ways for linking with Ansible

	smartcs_tty_command	Linking with network device vendor modules
Features	- Our original module for SmartCS - Need to define the console input/output results in Playbook like a TeraTerm macro	- Able to execute network device vendor modules via SmartCS (via console)
Advantages	- Managed nodes without Ansible module also can be placed under automation control. - Can be applied to works that are difficult to achieve with existing modules (reboot, verup, etc.) - Integrates all operations into a single Playbook.	- Existing **Playbook can be reused** *Switch just by editing "vars" - Idempotence is guaranteed
Cautions	- No guarantee of idempotence - Input/output information is required to create Playbook	- Only modules that support "network_cli" can be linked

SmartCS modules for Ansible

- Module explanation

ANSIBLE

Module explanation

SEIKO

■Modules provided as "SmartCS modules for Ansible"

[CLI Command Function]

Module	Description
smartcs_command	Executes the status display command or maintenance command to SmartCS and get the execution result. This module does not support the execution of setting commands. Use a "smartcs_config" module when setting SmartCS.
smartcs_config	Executes the setting command to SmartCS. Specify commands in "lines" or "src" by those displayed by "show config running" command.
smartcs_facts	Gathers the device information from SmartCS. The specifiable options are all, default, tty, and config.

[Console Access Function]

Module	Description
smartcs_tty_command	Sends the specified characters to the console of the devices connected to the SmartCS serial ports and gets the console input/output results.

SEIKO

■"smartcs_tty_command" module policies

Initial status of console

The "smartcs_tty_command" module does not manage or control the status of the managed nodes console.
Depending on the command which was last executed, the status of the managed nodes console may be in various expected statuses:
- Login prompt status
- General user shell status
- Privileged user shell status
- Shell status for configuration

Create the Playbook by considering the status of the managed nodes console.

Input/output results of console

The "smartcs_tty_command" does not automatically determine whether an error occurred in the execution result for the CLI command executed on the console of the managed nodes.
If you wish to control the execution result (ok/failed) of the "ansible-playbook" command according to the result of the CLI command executed on the console, use the following options.
- error_recvchar_regex option
- error_detect_on_module option

SEIKO

■Options used with the "smartcs_tty_command"

Option name	Setting range	Description
tty	1 to 48	The SmartCS serial port number to send the characters to. It can also be specified in a list format such as "1-10".
cmd_timeout	1 to 7200	The time to wait to receive "recvchar" after sending the characters.
nl	cr/ lf/ crlf	The line feed code sent when "__NL__."
sendchar(src)		The list of characters to send to the specified tty. The characters are sent in order from the top of the list. Line feed codes and control characters can also be sent.
		[Option] __CTL__:hex Send one control character. WAIT and NOWAIT option can be specified with this option.
		[Option] __HEX__:hexs Send multiple control characters. NOWAIT option can be specified with this option.
		[Option] __WAIT__:sec An option which specifies "cmd_timeout" described above for each sent character.
		[Option] __NOWAIT__ Immediately sends the next characters without waiting for the characters specified in "recvchar".
		[Option] __NOWAIT__:sec Sends the next characters after the specified period of time has passed without waiting for the characters specified in "recvchar".
recvchar (recvchar_regex)		The list of characters (prompt, etc.) that are expected to be received after sending the characters. When any of the characters in the list are received, it sends the next characters. The expected characters can be described with a regular expression.

SEIKO

■Options used with the "smartcs_tty_command"

Option name	Setting range	Description
error_detect_on_sendchar	cancel / exec	Specify whether to send the next characters when an error occurs after sending the characters.
error_detect_on_module	ok / failed	Specify whether to set the "ansible-playbook" command execution result to "ok" or "failed" when an error occurs after sending the characters.
error_recvchar_regex		A list that describes with a regular expression the received characters that you wish to determine as an error after sending the characters.
ttycmd_debug	off / on / detail	Display the debug information after the process of sending and receiving the characters has ended.

SEIKO

■Options used with the "smartcs_tty_command"

Option name	Setting range	Description
initial_prompt		The characters which are expected to be received after sending the "initial_prompt_check_cmd". ("Login:" etc.)
initial_prompt_check_cmd		Specify the command to check the status of console before sending the characters. ("__NL__" (Line feed), etc.)
initial_prompt_check_cmd_timeout	1 to 30	Specify the time to wait until checking the received characters after sending the "initial_prompt_check_cmd".
escape_cmd		Specify the command to send when unable to receive the "initial_prompt". ("exit" etc.)
escape_cmd_timeout	1 to 30	Specify the time to wait until checking the received characters after sending the "escape_cmd".
escape_cmd_retry	0 to 8	Specify the number of times to retry sending the "initial_prompt_check_cmd" when the "initial_prompt" cannot be received after sending the "escape_cmd".

SEIKO

■Options used with the "smartcs_tty_command"

Option name	Setting range	Description
custom_response	boolean value	Specify whether to output the sent characters (execute_command) and received characters (response) in a separate format for each command specified in the "sendchar" option in addition to "stdout" and "stdout_lines".
custom_response_delete_nl	boolean value	Specify whether to delete lines with only the line feed code for the "custom_response" output.
custom_response_delete_lastline	boolean value	Specify whether to delete the last line of the response for the "custom_response" output. Of the characters specified in the "recvchar" option, it is possible to not have the received characters (primarily the network devices prompt) included in the response.

Module explanation

SEIKO

■Playbook example

```
---
- name: smartcs_tty_command sample
  hosts: smartcs
  gather_facts: no

  tasks:
  - name: “StartupConfig by Console”
    smartcs_tty_command:
      tty: 1
      nl: cr
      cmd_timeout : 5
      recvchar :
      - “Username:  ”
      - “Password: ”
      - “Catalyst3560> ”
      : Omitted
      sendchar:
      - __NL__
      - cisco
      - cisco
      - enable
      : Omitted

  vars:
  - ansible_command_timeout: 30
  - ansible_connection: network_cli
  - ansible_network_os: smartcs
  - ansible_user: smartcs-ansible
  - ansible_password: xxxxxxxx
```

■recvchar (recvchar_regex)
- Specify multiple characters (prompt, etc.) that are expected to be received after sending the command.
- When any of the specified characters are received, the next character in "sendchar" is sent.

■sendchar
- Set the characters to send to the specified tty.
- The characters are sent in order from the top of the list.

- ansible_command_timeout
  -> Since executing the command from the console, more processing time is required than the typical module.
  Therefore, the timeout value should be extended. (default:10s)

- ansible_connection
  -> Specify "network_cli"

- ansible_network_os
  -> Specify "smartcs"

- ansible_user , ansible_password
  -> Specify the login information for the extusr to log into SmartCS.

SEIKO

■Playbook execution result

Name	Description	Trigger	Type
stdout	Command execution result	When the command execution is successful	List
stdout_lines	List of command execution results separated by each sent character		List

stdout output example

```
 "stdout": [
       "show version¥n¥nSystem: System Software Ver 2.0 (Build 2019-
03-25)¥n¥nBoot Status: Power on (00:01:00)¥n¥nSystem Up Time:
2019/05/22 15:33:07¥n¥nMain System: Ver 2.0¥n¥nBackup System: Ver 1.2
¥n¥n(c)NS-2250#",
],
```

stdout_lines output example

```
 "stdout_lines": [
       [
          "show version",
          "",
          "System          : System Software Ver 2.0 (Build 2019-03-25)",
          "",
          "Boot Status     : Power on (00:01:00)",
          "",
          "System Up Time  : 2019/05/22 15:33:07",
          "",
          "Main System     : Ver 2.0",
          "",
          "Backup System   : Ver 1.2",
          "",
          "(c)NS-2250#"
        ],

 ],
```

SEIKO

■Playbook execution result

Name	Description	Trigger	Type
stdout_lines_custom	For the characters sent and received by the console, output a list in a format that the sent characters (execute_command) and the received characters (response) are separated.	When the "custom_response" setting is enabled and the command execution is successful	List

Optional setting value

- custom_response : on
  =>Enable output in "stdout_lines_custom"
- custom_response_delete_nl : on
  =>Delete the spaces between lines in the command execution result
- custom_response_delete_lastline : off
  =>Not delete the last line (prompt, etc.)

Output example

```
"stdout_lines_custom": [
        {
            "execute_command": "show version",
            "response": [
                "System          : System Software Ver 2.0 (Build 2019-03-25)",
                "Boot Status     : Power on (00:01:00)",
                "System Up Time  : 2019/05/22 15:33:07",
                "Main System     : Ver 2.0",
                "Backup System   : Ver 1.2",
                "(c)NS-2250#"
            ]
        },
    ]
```

■Characters that can be sent with "sendchar"

・The sendable characters include all of the visual characters similar to "recvchar".

- sendchar

```
SPACE ! “ # $ % & ‘ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ¥ ] ^ _ `
a b c d e f g h I j k l m n o p q r s t u v w x y z { | } ~
```

- In module v1.0, the red characters and symbols above cannot be sent with the "sendchar" option. In module v1.1 and above, all of the visual characters can be sent similar to "recvchar".
- When specifying some symbols with "sendchar", they must be enclosed within single or double quotation marks.
  ’ (single quotation), ”(double quotation) etc.

- recvchar

```
SPACE ! “ # $ % & ‘ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ¥ ] ^ _ `
a b c d e f g h I j k l m n o p q r s t u v w x y z { | } ~
```

SEIKO

■Characters that can be sent with "sendchar"

・One control character can be sent with sendchar "__CTL__:hex" option.

Sendable control characters

00 : [Ctrl-@]	10 : [Ctrl-P]
01 : [Ctrl-A]	11 : [Ctrl-Q]
02 : [Ctrl-B]	12 : [Ctrl-R]
03 : [Ctrl-C]	13 : [Ctrl-S]
04 : [Ctrl-D]	14 : [Ctrl-T]
05 : [Ctrl-E]	15 : [Ctrl-U]
06 : [Ctrl-F]	16 : [Ctrl-V]
07 : [Ctrl-G]	17 : [Ctrl-W]
08 : [Ctrl-H]	18 : [Ctrl-X]
09 : [Ctrl-I]	19 : [Ctrl-Y]
0a : [Ctrl-J]	1a : [Ctrl-Z]
0b : [Ctrl-K]	1b : [Ctrl-[]
0c : [Ctrl-L]	1c : [Ctrl-¥]
0d : [Ctrl-M]	1d : [Ctrl-]]
0e : [Ctrl-N]	1e : [Ctrl-^]
0f : [Ctrl-O]	1f : [Ctrl-_]
	7f : [Delete]

Playbook

```
sendchar:
  - show version
  - ping count 1000 172.31.1.1
  - __CTL__:03
```

- Supports each send option similar to "sendchar"

```
__CTL__:03
__CTL__:03 __WAIT__:30
 => wait the specified time for the recvchar return after sending
__CTL__:03 __NOWAIT__:30
 => wait the specified time after sending (do not wait for recvchar)
__CTL__:03 __NOWAIT__
 => send the next sendchar immediately after sending
```

SEIKO

■Characters that can be sent with "sendchar"

・Multiple control characters can be sent with sendchar "__HEX__:hexs" option.

Sendable control characters

ASCII 00 to 7F can be sent

*SmartCS doesn't wait the characters that are specified in recvchar when sending the characters with "__HEX__".

If it's necessary to wait them "__WAIT__" option should be specified after "__HEX__" option.

Playbook

```
sendchar:
  - sudo nmtui __NOWAIT__
  - __HEX__:1b 5b 42
  - __HEX__:0d
  - __HEX__:08 08 08 08
  - __HEX__:54 65 73 74
  - __HEX__:1b 5b 42
  - __HEX__:0d
  - exit
```

- Supports each send option similar to "sendchar"

```
__HEX__:0d
__HEX__:0d __WAIT__:30
 => wait the specified time for the recvchar return after sending
__HEX__:0d __NOWAIT__:30
 => wait the specified time after sending (do not wait for recvchar)
__HEX__:0d __NOWAIT__
 => send the next sendchar immediately after sending
```

SEIKO

■Specify "sendchar" in "src"

・An external file can be specified for the characters to send instead of "sendchar".

Playbook

```
recvchar_regex:
- '[Uu]sername: '
- '[Ll]ogin: '
- '[Pp]assword: '
- '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#)'
src: sendchar_cisco.txt
```

External file

```
__NL__
ssol
ssol
terminal length 0
show version
show vlan brief
exit
```

sendchar_cisco.txt

・Caution

- Only the "sendchar" or "src" options may be specified (exclusive setting)

SEIKO

■Specify "sendchar" in "src"

・Playbook example using the "src" option

Playbook

```
vars:
- cs_parameters:
  - { tty: '4', config: ./src/config4 }
  - { tty: '5', config: ./src/config5 }

tasks:
- name: "smartcs_tty_command"
  smartcs_tty_command:
    tty: '{{ item.tty }}'
    recvchar:
    - 'Username: '
    - 'Password: '
    - 'Press RETURN to get started.‘
    recvchar_regex:
    - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#)'
    src: '{{ item.config }}'
  with_items: '{{ cs_parameters }}'
```

・Specify cs_parameters (dict type) as vars
*Also possible even when defined in group_vars, host_vars

・Prepare tty and config as keys within dict and specify each parameter

tty information

src information
*sendchar

Processing flow

For one execution of the Playbook, the "smartcs_tty_command" task processing is executed referencing the values of the "cs_parameters".
*recvchar is common

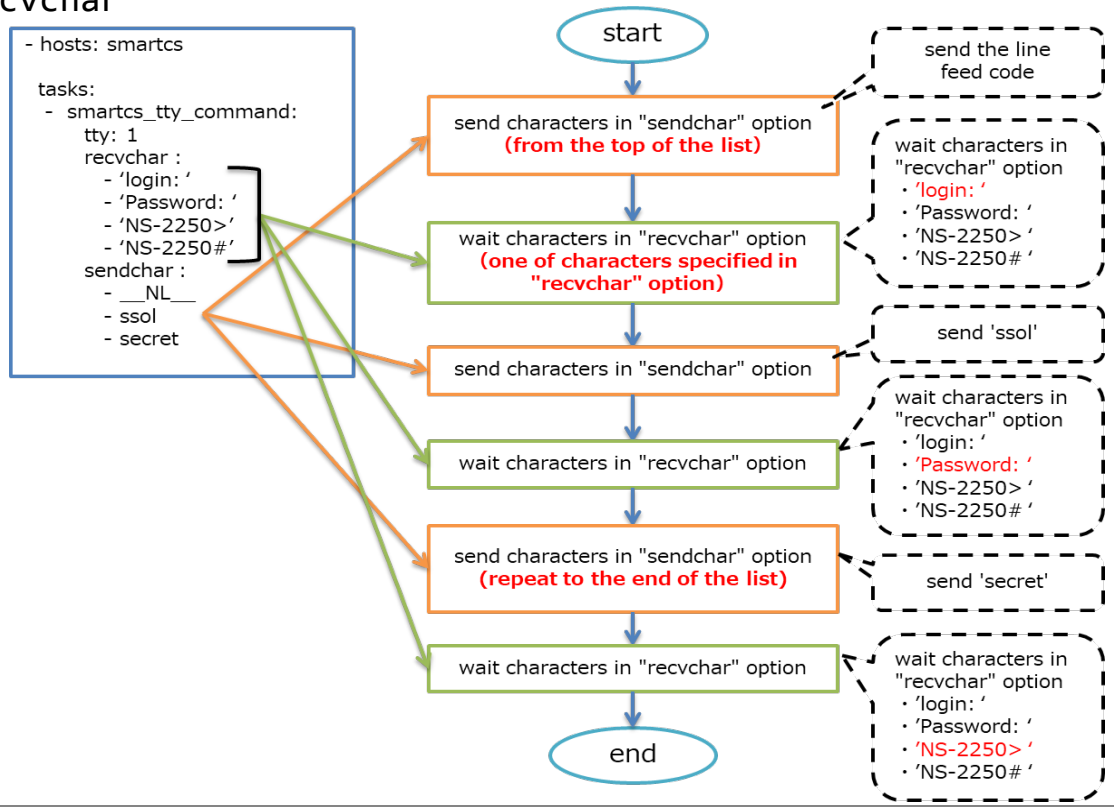
```
(1) smartcs_tty_command
  - tty: '4'
  - src: './src/config4'
```

```
(2) smartcs_tty_command
  - tty: '5'
  - src: './src/config5'
```

SEIKO

■ "smartcs_tty_command" option

sendchar / recvchar

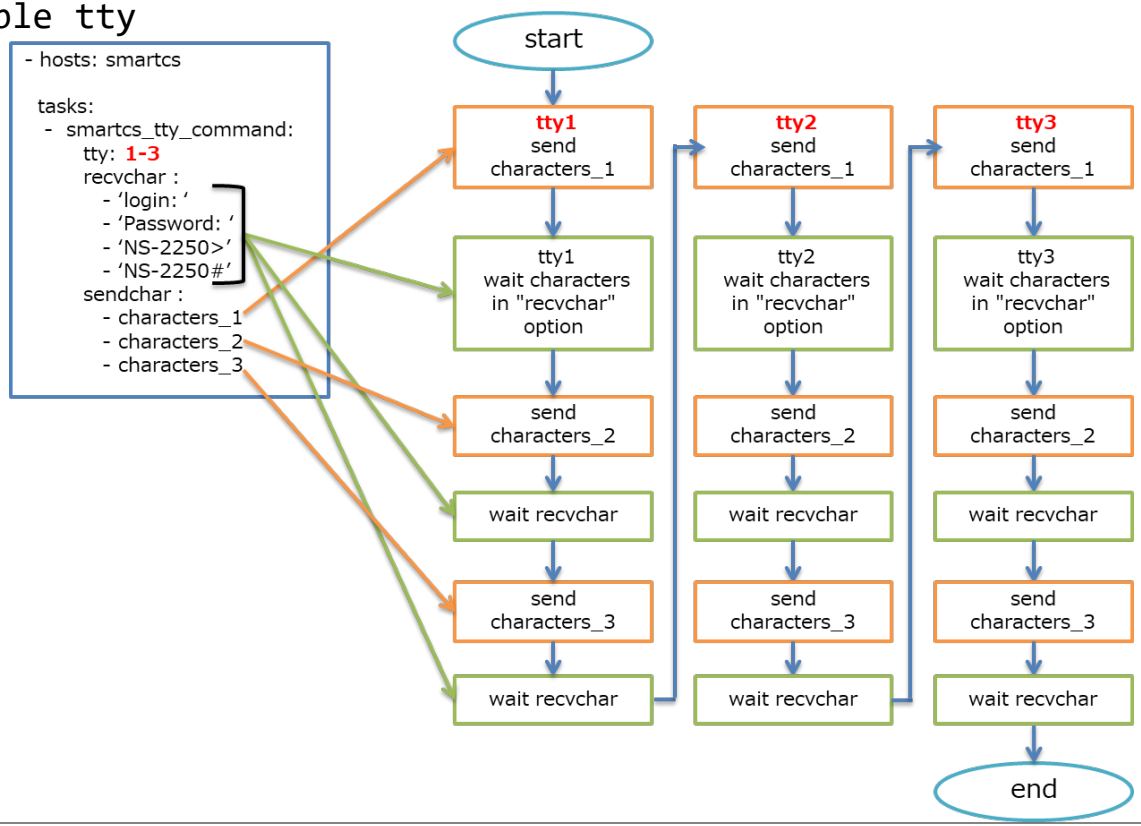


SEIKO

■ "smartcs_tty_command" option

Specify multiple tty

```
- hosts: smartcs

  tasks:
   -  smartcs_tty_command:
        tty: 1-3
        recvchar :
          - 'login: '
          - 'Password: '
          - 'NS-2250>'
          - 'NS-2250#'
        sendchar :
          - characters_1
          - characters_2
          - characters_3
```

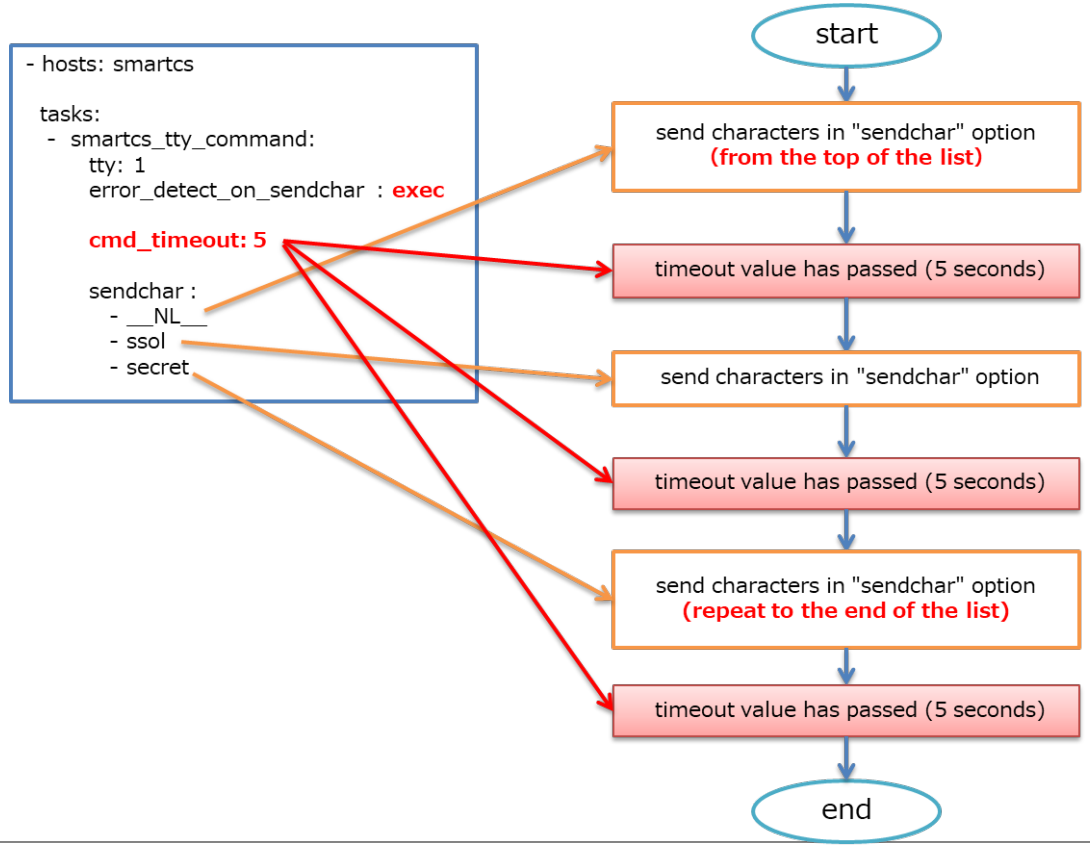


SEIKO

■ "smartcs_tty_command" option

cmd_timeout

```
- hosts: smartcs

  tasks:
   -  smartcs_tty_command:
        tty: 1
        error_detect_on_sendchar : exec

        cmd_timeout: 5

        sendchar :
          - __NL__
          - ssol
          - secret
```

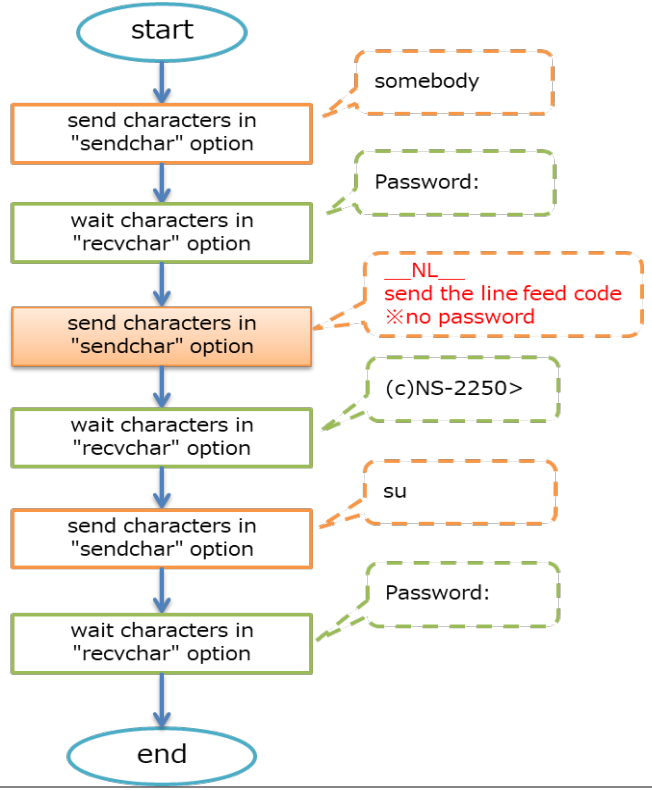


SEIKO

■ "smartcs_tty_command" option

Operation of the option "__NL__" in "sendchar"

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - somebody
         - __NL__
         - su
         - __NL__
         - show version
```



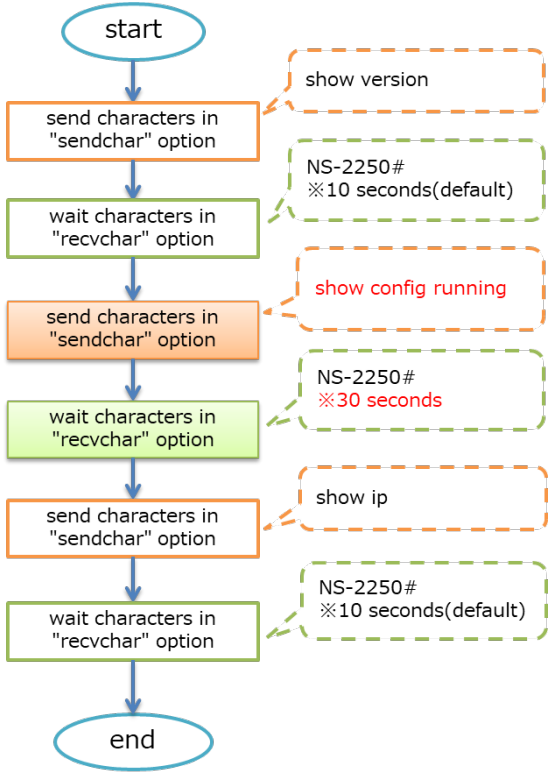
Use when you wish to send only line feeds

SEIKO

■"smartcs_tty_command" option

Operation of the option "__WAIT__:Xsec" in "sendchar"

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - show version
         - show config running __WAIT__:30
         - show ip
```



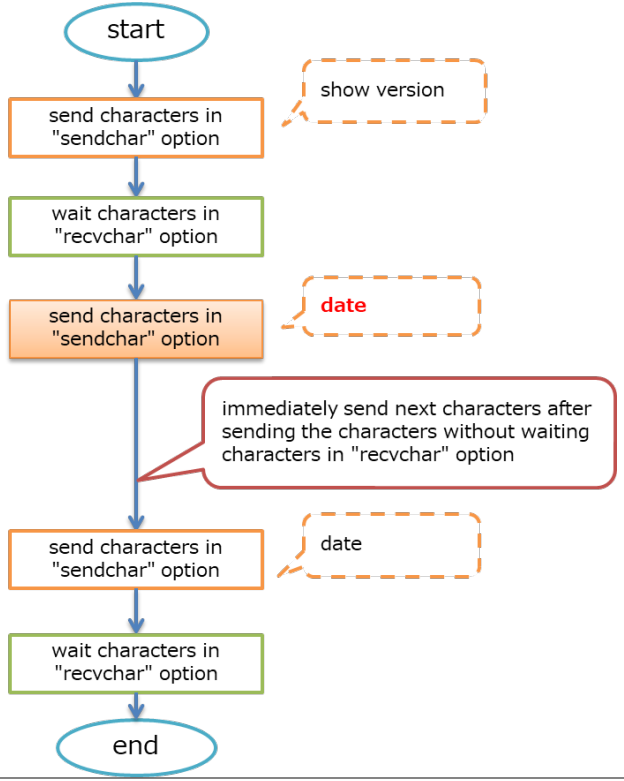
You can use this option when you wish to change the timeout period after sending a specific "sendchar".
*When only the specific command has a long execution time, etc.

SEIKO

■ "smartcs_tty_command" option

Operation of the option "__NOWAIT__" in "sendchar"

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       cmd_timeout: 5
       recvchar :
         - 'login: '
         - 'Password: '
         - '>'
         - '#'
         - '[y/n] ? '
       sendchar :
         - show version
         - date __NOWAIT__
         - date
```



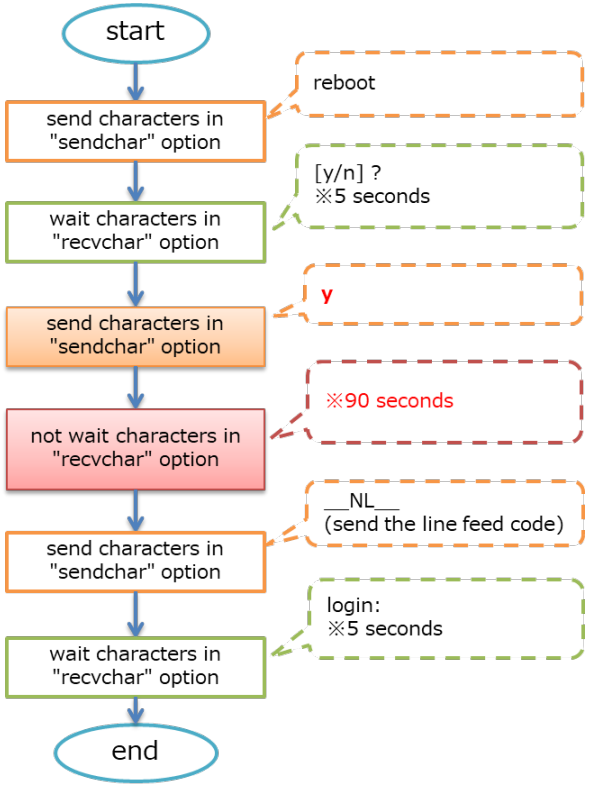
This option can be used when you wish to sequentially send characters without waiting for "recvchar" after sending "sendchar".

SEIKO

■ "smartcs_tty_command" option

Operation of the option "__NOWAIT__:Xsec" in "sendchar"

```
- hosts: smartcs

  tasks:
   -  smartcs_tty_command:
        tty: 1
        cmd_timeout: 5
        recvchar :
          - 'login: '
          - 'Password: '
          - '>'
          - '#'
          - '[y/n] ? '
        sendchar :
          - show version
          - reboot
          - y __NOWAIT__:90
          - __NL__
          - show version
```



This option can be set the amount of time not waiting for the set "recvchar" after sending a specific "sendchar".
*During a reboot or other times when you receive a "recvchar" in an unintended section.

SEIKO

■"smartcs_tty_command" option

In some cases, sending the characters set in "sendchar" may result in an error.

Causes of errors which occur after sending "sendchar"		Output
Unable to receive "recvchar" before the end of the timeout period		Error:: Timeout.
Unable to connect to the target tty	Unable to connect, because there is no access permission setting	Error:: Not allowed.
	Unable to connect due to exclusive control	Error:: Session limit over.
	Unable to connect to the tty management daemon	Error:: Connection closed.
	Detected the characters set in "error_recvchar_regex"	Error:: Matched “xxxxx”.
Not to send the next "sendchar" when "error_detect_on_sendchar" is set to "cancel"		Error:: After error.

*When no permission to access
If the Extusr group user does not have the appropriate authority or if the tty management function is not enabled, it will result in an error.

*About exclusive processing
Access via Ansible (access through the tty management function) and access through a conventional port user cannot be performed at the same time. The first connection takes priority.

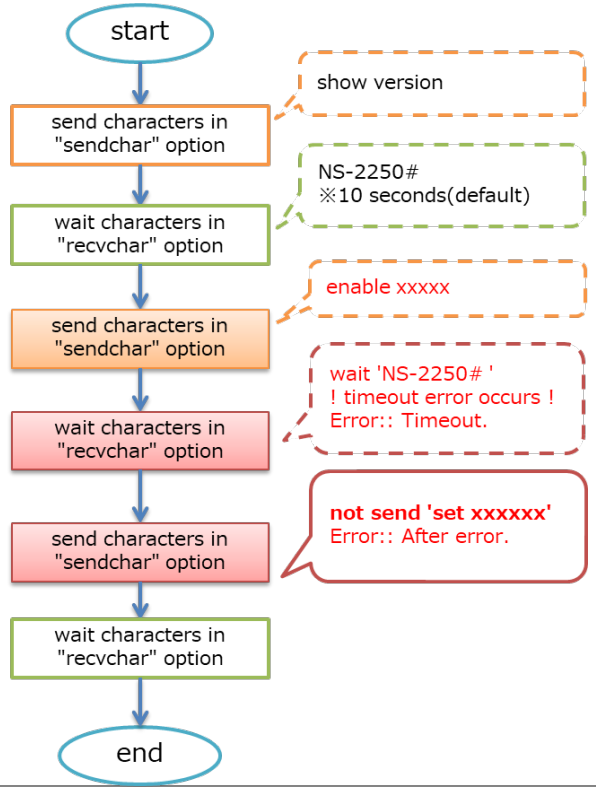
*About "error_recvchar_regex"
Disabled when not set.
When it is set and the specified characters are included in the characters sent and received, it is determined to be an error.

SEIKO

■"smartcs_tty_command" option

Operation when the "error_detect_on_sendchar" option is set to "cancel"

```
- hosts: smartcs

  tasks:
   - smartcs_tty_command:
       tty: 1
       error_detect_on_sendchar: cancel
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - show version
         - enable xxxxx
         - set xxxxxx
```



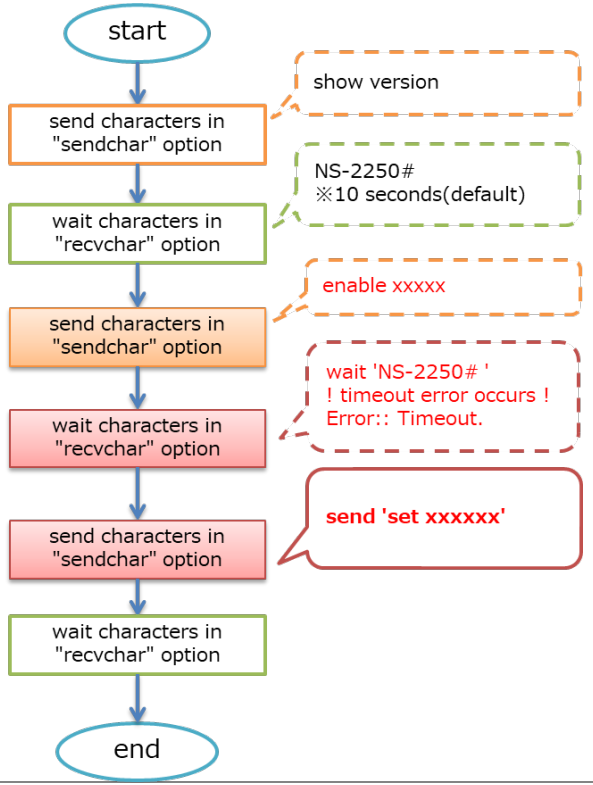
This option is set to not send "sendchar" after an error has occurred.
*Malfunction prevention, etc.

SEIKO

■ "smartcs_tty_command" option

Operation when the "error_detect_on_sendchar" option is set to "exec"

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       error_detect_on_sendchar: exec
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - show version
         - enable xxxxx
         - set xxxxxx
```



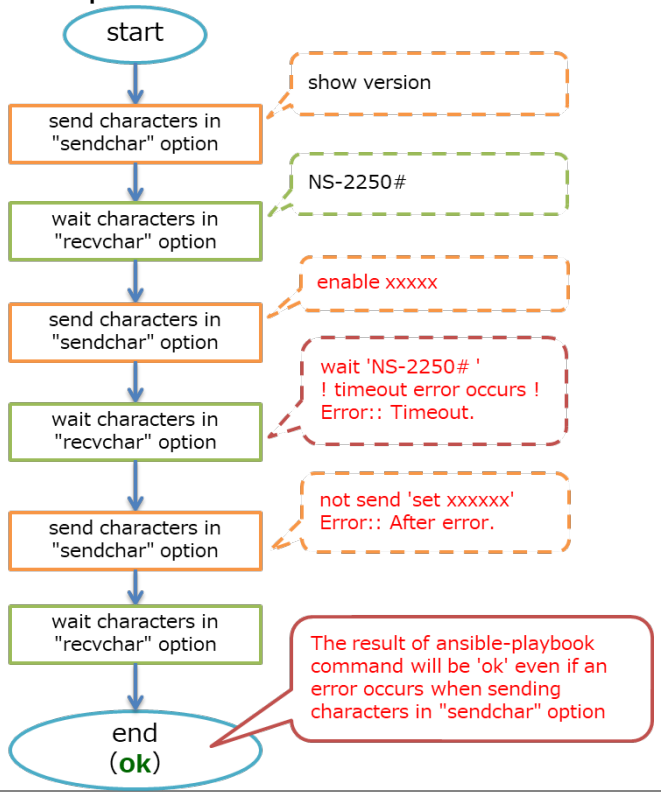
This option is set to send "sendchar" even when an error has occurred.

SEIKO

■"smartcs_tty_command" option

Operation when the "error_detect_on_module" option is set to "ok"

```
- hosts: smartcs

  tasks:
   -  smartcs_tty_command:
        tty: 1
        error_detect_on_sendchar: cancel
        error_detect_on_module: ok
        recvchar :
          - 'login: '
          - 'Password: '
          - 'NS-2250>'
          - 'NS-2250#'
        sendchar :
          - show version
          - enable xxxxx
          - set xxxxxx
```



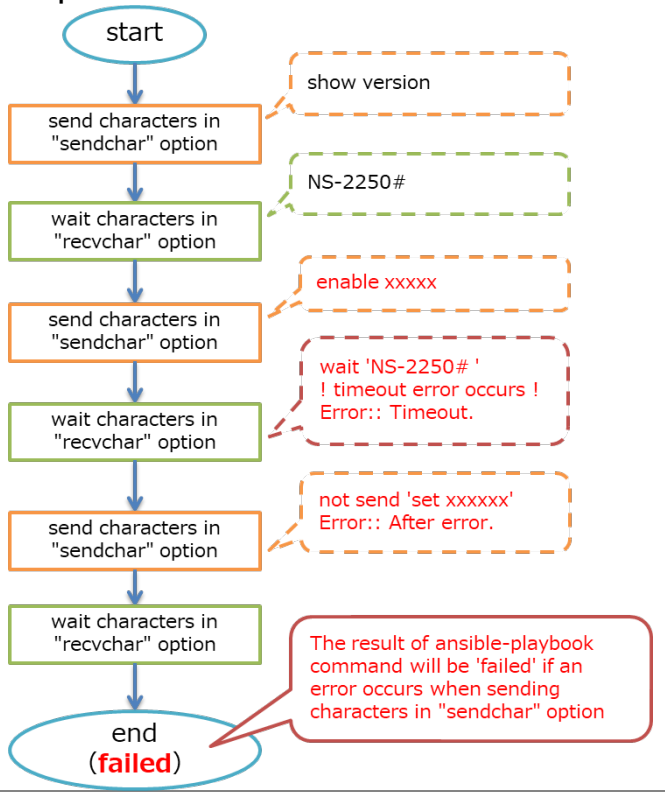
This option sets an ansible command to "ok" rather than "failed" even when an error has occurred.
*Errors that can be determined are limited to the "causes of errors which occur after sending sendchar."

SEIKO

■"smartcs_tty_command" option

Operation when the "error_detect_on_module" option is set to "failed"

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       error_detect_on_sendchar: cancel
       error_detect_on_module: failed
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - show version
         - enable xxxxx
         - set xxxxxx
```



This option sets an ansible command to "failed" when an error has occurred.
*Errors that can be determined are limited to the "causes of errors which occur after sending sendchar."
*Linking with Ansible Tower and AWX, etc.

SEIKO

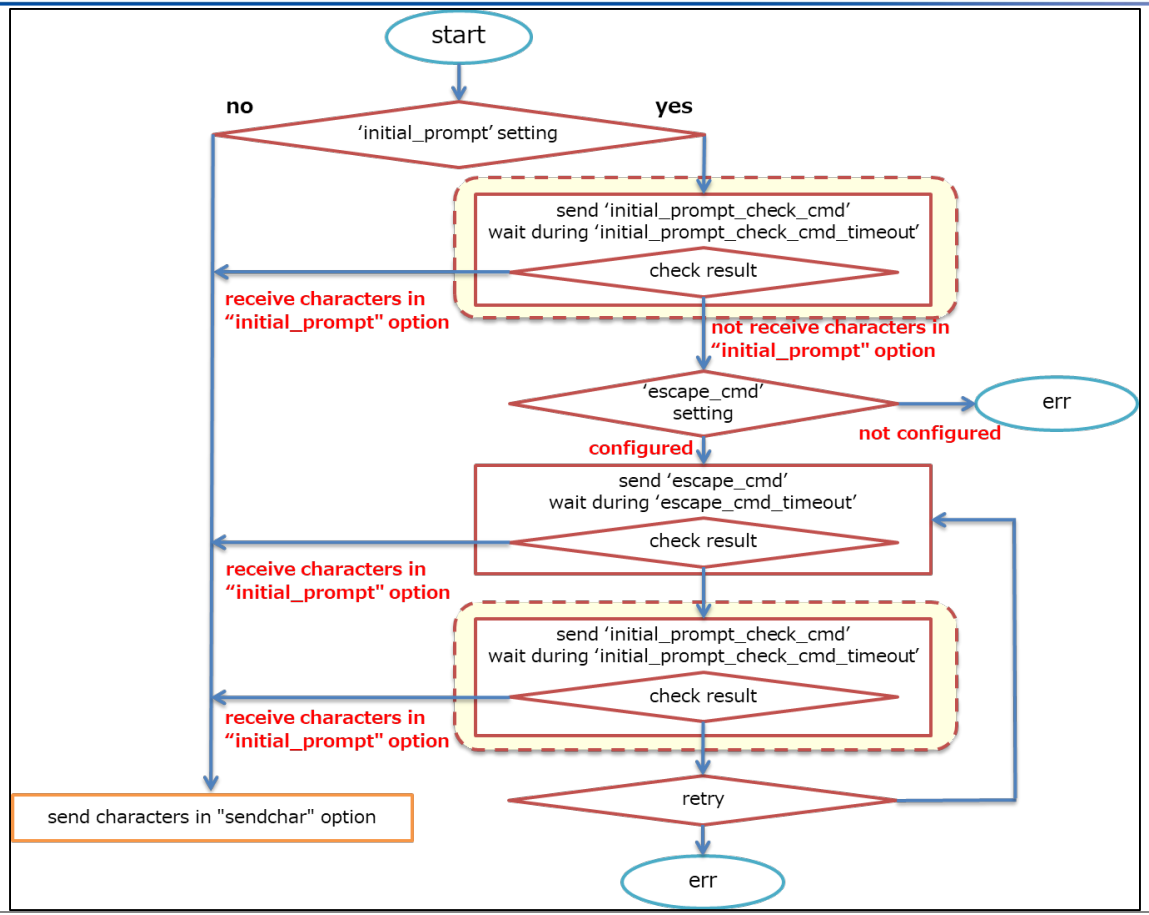
■ Function for checking the status of console before sending "sendchar"

・Check whether the status of console is the expected status before sending "sendchar".

Playbook

```
smartcs_tty_command:
  tty: '15'
  initial_prompt: 'User Access Verification'
  initial_prompt_check_cmd: '__NL__'
  initial_prompt_check_cmd_timeout: 3
  escape_cmd_timeout: 3
  escape_cmd: 'exit'
  recvchar:
    - 'Press RETURN to get started.'
  recvchar_regex:
    - '[Uu]sername: '
    - '[Pp]assword: '
    - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#)'
  sendchar:
    - __NL__
    - userA
    - secret
    - terminal length 0 __WAIT__:5
```

Option name	Description
initial_prompt	Set the expected characters. This option can also be set with a regular expression.
initial_prompt_check_cmd	Specify the command to check the status of console before sending "sendchar". The default value is a line feed (__NL__).
initial_prompt_check_cmd_timeout	Set the timeout value after sending the check command. (default 3s)
escape_cmd	Set the command to send when the expected characters are not output. Example (exit, logout, etc.)
escape_cmd_timeout	Set the timeout value of the "escape_cmd". (default 3s)
escape_cmd_retry	Set the number of retries. (default 3)



SEIKO

■"smartcs_tty_command" return value extension

・Options which add a return value (stdout_lines_custom) that outputs the console input/output in an easy to understand format other than "stdout" and "stdout_lines".

Playbook

```
smartcs_tty_command:
  tty:1
  cmd_timeout: 10
  custom_response: on
  custom_response_delete_nl: on
  custom_response_delete_lastline: on
  recvchar_regex:
    - '[Uu]sername: '
    - '[Pp]assword: '
    - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#)'
  sendchar:
    - 'somebody'
    - __NL__
    - show version
```

Option name	Description
custom_response	Sends a return value in a format which can separate between "sendchar" and "recvchar" in addition to "stdout" and "stdout_lines". (stdout_lines_custom) Outputs "execute_command" and "response" separately for each "sendchar".
custom_response_delete_nl	Deletes the line feed only line in the "custom_response" output.
custom_response_delete_lastline	Deletes the last line of the response in the "custom_response" output. *The purpose of this option is to not display the prompt after executing a CLI command. *Outputs only show related command execution results

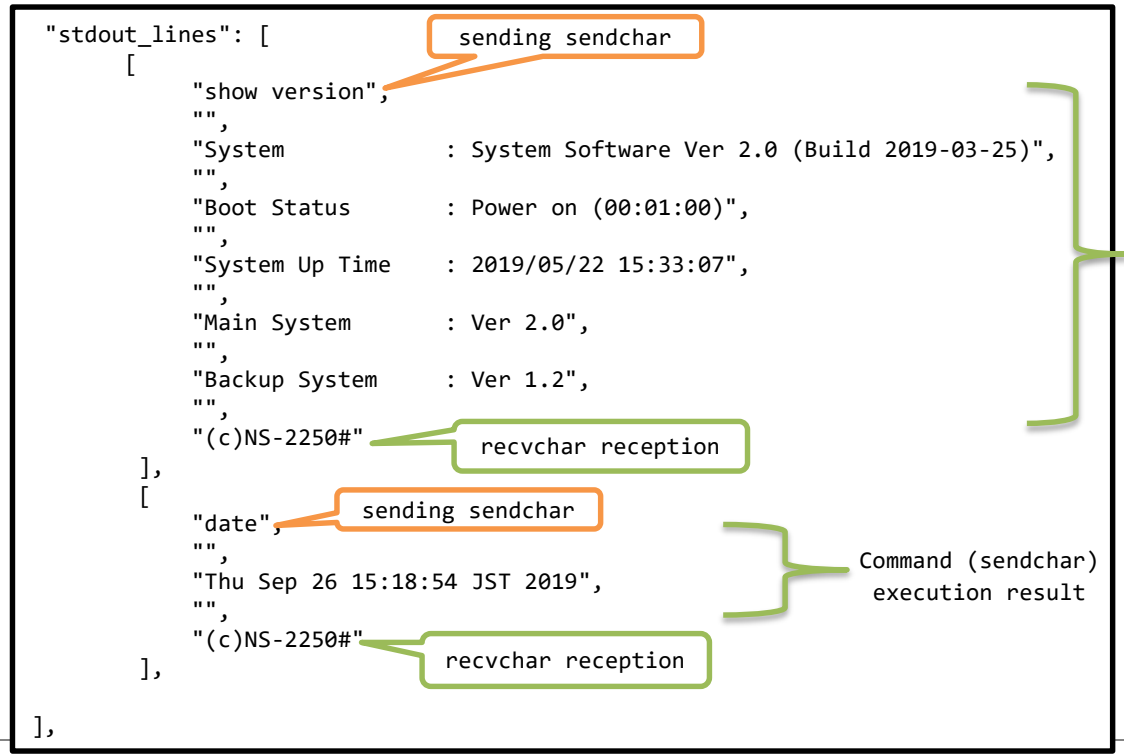
SEIKO

■"smartcs_tty_command" return value extension

Setting example

- custom_response : off
- custom_response_delete_nl : off
- custom_response_delete_lastline : off

Display example <Same as the conventional "stdout_lines">



SEIKO

■"smartcs_tty_command" return value extension

Setting example

- custom_response : **on**
- custom_response_delete_nl : off
- custom_response_delete_lastline : off

Display example <"stdout_lines_custom" extended in v1.1>

```
" stdout_lines_custom": [
      {
      "execute_command" : "show version",
      "response“ : [
              "",
              "System            : System Software Ver 2.0 (Build 2019-03-25)",
              "",
              "Boot Status       : Power on (00:01:00)",
              "",
              "System Up Time    : 2019/05/22 15:33:07",
              "",
              "Main System       : Ver 2.0",
              "",
              "Backup System     : Ver 1.2",
              "",
              "(c)NS-2250#"
          ]
       },
       {
      "execute_command" : "date",
      "response“ : [
              "",
              "Thu Sep 26 15:18:54 JST 2019",
              "",
              "(c)NS-2250#"
          ],
       }
]
```

sending sendchar

Command (sendchar) execution result

recvchar reception

sending sendchar

Command (sendchar) execution result

recvchar reception

SEIKO

■"smartcs_tty_command" return value extension

Setting example

- custom_response : **on**
- custom_response_delete_nl : **on**
- custom_response_delete_lastline : off

Display example <"stdout_lines_custom" extended in v1.1>

```
" stdout_lines_custom": [
      {
      "execute_command" : "show version",
      "response“ : [
              "System            : System Software Ver 2.0 (Build 2019-03-25)",
              "Boot Status       : Power on (00:01:00)",
              "System Up Time    : 2019/05/22 15:33:07",
              "Main System       : Ver 2.0",
              "Backup System     : Ver 1.2",
              "(c)NS-2250#"
          ]
       },
       {
      "execute_command" : "date",
      "response“ : [
              "Thu Sep 26 15:18:54 JST 2019",
              "(c)NS-2250#"
           ],
        }
]
```

Delete the blank lines from the execution result after sending "sendchar"

->Consider the visibility of the command execution result and the effort during extraction
*Reduce the number of formatting steps performed on the Playbook side

SEIKO

■"smartcs_tty_command" return value extension

Setting example

- custom_response : **on**
- custom_response_delete_nl : **on**
- custom_response_delete_lastline : **on**

Delete the last line (prompt) from the execution result after sending "sendchar"

->Format it so that it has only the command execution result.
*Reduce the number of formatting steps performed on the Playbook side

Display example <"stdout_lines_custom" extended in v1.1>

```
" stdout_lines_custom": [
      {
      "execute_command" : "show version",
      "response“ : [
              "System            : System Software Ver 2.0 (Build 2019-03-25)",
              "Boot Status       : Power on (00:01:00)",
              "System Up Time    : 2019/05/22 15:33:07",
              "Main System       : Ver 2.0",
              "Backup System     : Ver 1.2",
          ]
       },
       {
      "execute_command" : "date",
      "response“ : [
              "Thu Sep 26 15:18:54 JST 2019",
           ],
        }
]
```

SEIKO

■"smartcs_tty_command" return value extension

Playbook excerpt

```
    sendchar:
      - somebody
      - __NL__
      - su
      - __NL__
      - show version __WAIT__:10
      - exit
      - exit
    register: result

- name: execute_command
  debug:
    msg: "{{ result.stdout_lines_custom[4].execute_command }}"

- name: response
  debug:
    msg: "{{ result.stdout_lines_custom[4].response }}"
```

```
TASK [execute_command] ***************************************************
task path: /home/nsxi/smartcs/playbook/precheck_cs.yml:41
ok: [172.31.8.16] => {
    "msg": "show version "
}

TASK [response] ***********************************************************
task path: /home/nsxi/smartcs/playbook/precheck_cs.yml:45
ok: [172.31.8.16] => {
    "msg": [
        "System                : System Software Ver 2.0 (Build 2019-03-25)",
        "Boot Status           : Power on (00:01:00)",
        "System Up Time        : 2019/05/22 15:33:07",
        "Local MAC Address     : 00:80:15:42:00:08",
        "Number of MAC Address : 2",
        "Model                 : NS-2250-16 (16 port)",
        "Serial No.            : 56000050",
        "BootROM               : Ver 1.0",
        "Main Board CPU        : e500v2 (533.333328MHz)",
        "Main Memory           : 1025264 KBytes",
        "Boot System           : main (Ver 2.0)",
        "Boot Config           : internal startup1",
        "Main System           : Ver 2.0",
        "Backup System         : Ver 1.2",
    ]
}
```

SmartCS modules for Ansible

- Required settings for SmartCS

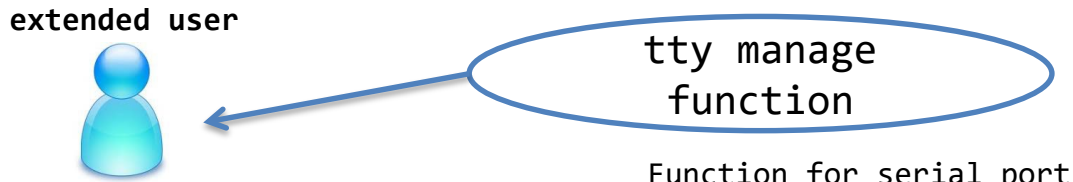
ANSIBLE

Required settings for SmartCS

SEIKO

■ To Use the "smartcs_tty_command" module

- Create an extended user (extusr group)
- Grant authority for the tty manage function to the extended user (extusr group)
- The tty manage function must be enabled.



Function for serial port access

Same authority as a "general user" just after creation
*Can become a privileged user with the "su" command

By logging in as extended user, a user with an extended executable CLI becomes available

Required settings for SmartCS

SEIKO

■Using the "smartcs_tty_command" module

- Create an extended user (extusr group)

```
create user <username> group extusr port <port_number> password
```

  - Create an extended user (extusr group) that can use the "smartcs_tty_command".
  - An accessible serial port number and password must also be set.
  - The user name/password to set correspond to the user name specified in "ansible_user" and the password specified in "ansible_password" when accessing from Ansible.

- Grant authority to the created user

```
set user <username> permission ttymanage on
```

  - Grant authority for the tty manage function to the extended user (extusr group).

- Enable the function

```
enable ttymanage
```

  - Enable the tty manage function.

Required settings for SmartCS

SEIKO

■Extended user (extusr group) overview

User group	Group name	Authority					
		Status statistical information display	Device settings	Telnet/SSH login to the device	FTP/SFTP login to the device	Login to the device console port	Access to the managed device (serial port)
Privileged User	root	○	○	○	×	○	×
General User	normal	○	×	○	×	○	×
Entended user	**extusr**	○	×	○*1	×	×	○*2
Port User (access to the serial port)	portusr	×	×	×	×	×	○
FTP/SFTP User	setup verup log	×	×	×	○	×	×

*1 The extended user can login to SmartCS only via SSH access.

*2 The extended user uses the CLI command (tty manage function) as the method to access the devices connected to SmartCS.

■Other

- Number of simultaneous extended user connections: 48 sessions
- Operate according to the sshd object settings.
  (Authentication method, port number, allowhost and ipfilter)

Available in system
v2.1 and above

Linking with network device vendor modules

- About the linking function
- Required settings for SmartCS

ANSIBLE

Linking with network device vendor modules

- Linking function with network device vendor modules
  - Regarding Playbooks created to operate Cisco, Arista or other devices, normally the process is executed by connecting via SSH, but it can be executed from the SmartCS console.
  - The Playbook task section can be reused without any changes.

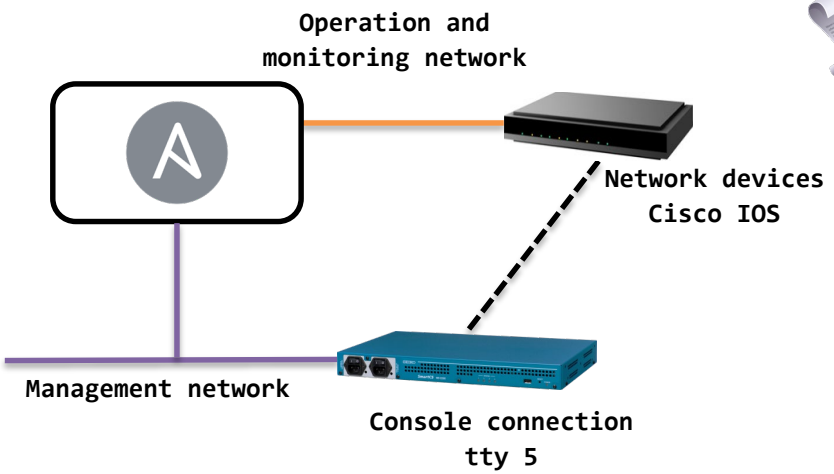
Configuration

Linking with network device vendor modules

SEIKO

■ Representation of linking with network device vendor modules

Configuration



Playbook

```
---
- hosts: smartcs
  gather_facts: no
  tasks:
  - name: "Task ios_command"
    ios_command:
      commands:
      - show version
      - show interfaces
      - show arp
      - show ip route

  vars:
    ansible_connection: network_cli
    ansible_user: port
    ansible_password: port
    ansible_ssh_port: 9305
    ansible_network_os: ios
    ansible_become: yes
    ansible_become_method: enable
    ansible_become_password:secret
    ansible_command_timeout: 60
```

Connection destination is SmartCS

Specify the network device vendor module

Specify network_cli for the connection plugin

Specify the SmartCS port user for the user name and password

Change the SSH connection port

Specify the appropriate vendor OS

Linking with network device vendor modules

SEIKO

■ Execution of network device vendor modules (Playbook configuration example)

Playbook (1)
Module: smartcs_tty_command

User: Extended User
Port: SSH port (22)

Login process to the device console via SmartCS

Playbook (2)
Module: network device vendor module

User: Port User
Port: sshxpt port (93xx)

Device operation (setting, display) via SmartCS

Playbook (3)
Module: smartcs_tty_command

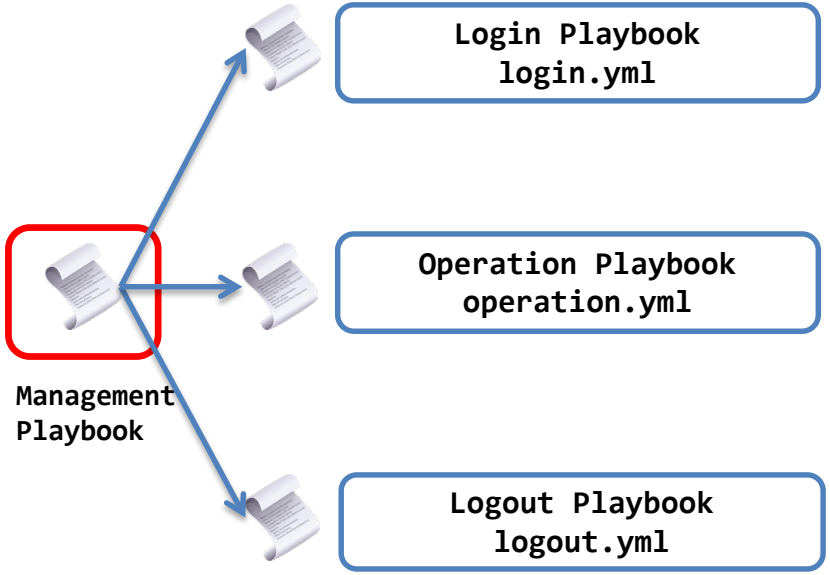
User: Extended User
Port: SSH port (22)

Logout process from the device console via SmartCS

Linking with network device vendor modules

SEIKO

■ Execution of network device vendor modules (Playbook configuration example)



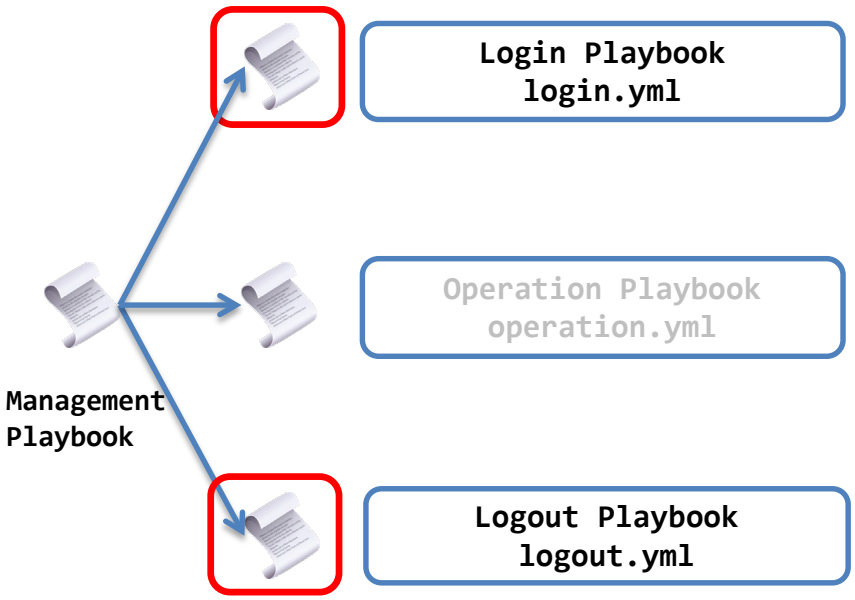

```
---
- name: "LOGIN with smartcs_tty_command"
  import_playbook: login.yml

- name: "Exec Task with ios_command"
  import_playbook: operation.yml

- name: "LOGOUT with smartcs_tty_command"
  import_playbook: logout.yml
```

Management Playbook example
*Actual Playbook to execute

Linking with network device vendor modules

SEIKO

■ Execution of network device vendor modules (Playbook configuration example)



login.yml

```
---
- hosts: smartcs
  tasks:
  - name: "Login by Console"
    smartcs_tty_command:
      tty: 5
      recvchar_regex:
      - '[Uu]sername: '
      - '[Pp]assword: '
      - '(^|¥r|¥n|!)[a-zA-Z0-9_.-]*(>|#)'
      sendchar :
      - __NL__
      - ios_user     <-Login ID for the iOS device
      - secret       <-Login password for the iOS device
```

Specify the login prompt of the console according to the network device

Example of a general-purpose network device prompt

Specify the user name and password when logging in to the device console

logout.yml

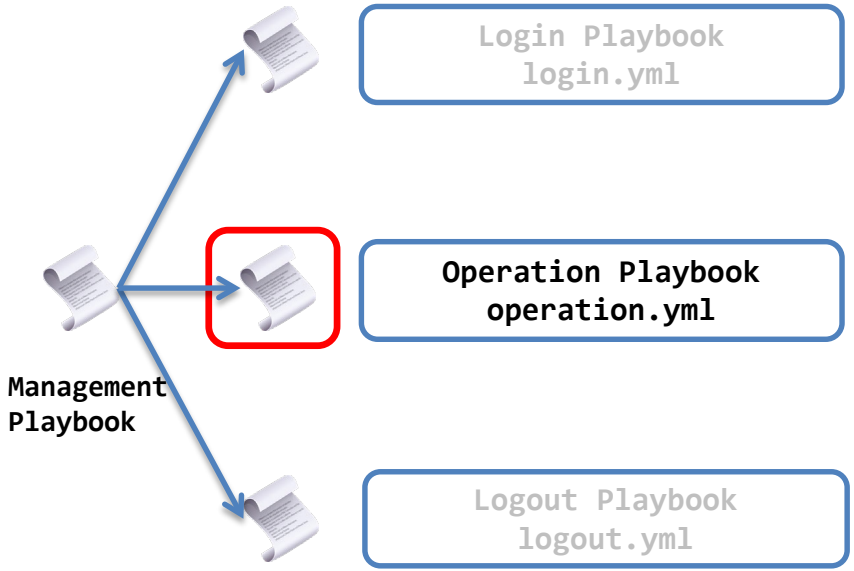
```
---
- hosts: smartcs
  tasks:
  - name: "Logout by Console"
    smartcs_tty_command:
      tty: 5
      recvchar :
      - "Press RETURN to get started.“
      recvchar_regex:
      - '(^|¥r|¥n|!)[a-zA-Z0-9_.-]*(>|#)'
      sendchar :
      - exit
```

Specify multiple exit transmissions according to the network device

Linking with network device vendor modules

SEIKO

■ Execution of network device vendor modules
(Playbook configuration example)



operation.yml

```
---
- hosts: smartcs
  gather_facts: no
  tasks:
  - name: "Task ios_command"
    ios_command:
      commands:
      - show version
      - show interfaces
      - show arp
      - show ip route

  vars:
    ansible_connection: network_cli
    ansible_user: port
    ansible_password: port
    ansible_ssh_port: 9305
    ansible_network_os: ios
    ansible_become: yes
    ansible_become_method: enable
    ansible_become_password:secret
    ansible_command_timeout: 60
```

Connection destination is SmartCS

Specify the network device vendor module

Specify network_cli for the connection plugin

Specify the SmartCS port user for the user name and password

Change the SSH connection port

Specify the appropriate vendor OS

Linking with network device vendor modules

SEIKO

■Execution of network device vendor modules

<Points to pay attention to when linking>

・Linking is limited to network device vendor modules that support "network_cli"

- ・Because of the internal process that the operation to login via SSH and execute the CLI is performed via console

  Example

```
vars:
  ansible_connection: network_cli
```

- ・It will not run if the prompt definition is different between the SSH connection and console access (terminal plugin definition)

・Pay attention to processing speed

- ・Network device vendor modules normally connect and run via SSH, but this linking runs via console Therefore, the processing speed is slow, so the timeout period must be extended. (command execution time, etc.)

  Example

```
vars:
  ansible_command_timeout: 60
```

Required settings for SmartCS

SEIKO

■ To link with network device vendor modules

・Prepare a new service port rather than using an existing TCP port

```
set portd tty <ttylist> session { both | telnet | ssh | none } { both | rw | ro } [ sshxpt ]
```

- Specify the "sshxpt" option to newly open TCP ports 9301 to 9348 and wait for a port connection.
- Because this port operates independently from existing direct/select service ports, it does not impact existing services.
- This port number corresponds to the number specified in the "ansible_port" when accessing from Ansible.

・The port starting number can be changed.

```
set portd sshxpt <port_num>
```

- Setting range: 1025 to 65000
- Default value: 9301

・Support for related display commands

- show portd , show portd tty

Required settings for SmartCS

SEIKO

■ To link with network device vendor modules

・Specify an action when accessed (line feed code transmission)

```
set portd tty <ttylist> connted send_nl { cr | crlf | lf | none }
```

- Specify a line feed code to be sent when accessed to the sshxpt port.
- The default value is "none" (not send anything even when accessed to the sshxpt port)

*A line feed code is sent when accessed and the prompt is output to run the "network_cli" plugin.

・Create a port user (portusr group)

```
create user <username> group portusr port <port_number> password
```

- Create a port user (portusr group) that can use the sshxpt function.
- An accessible serial port number and password must also be set.
- The user name and password to be set correspond to the user name specified in "ansible_user" and the password specified in "ansible_password" when accessing from Ansible using a network device vendor module.

Reference Information

- WEBINAR
- Ansible Automates Tokyo 2020
- Ansible Hands-on

ANSIBLE

Reference Information

SEIKO

■ Past lectures and Ansible Hands-on

- WEBINAR
  Starting "Fail-proof IT and Network Automation" with Ansible
  ~Importance of SmartCS in IT and Network Automation~

  http://redhat.lookbookhq.com/c/65-42?x=8XYa3o&lx=t84IoG

- Ansible Automates Tokyo 2020
  Role of SmartCS in Supporting Operation Automation and an Introduction to User Examples

  https://redhat.lookbookhq.com/automates-tokyo-2020/ssol-ansible-automat?lx=1ocUbB

- Ansible Hands-on
  - SmartCS x ALAXALA x Ansible Hands-on
  - SmartCS x IOS x Ansible Hands-on

  https://github.com/ssol-smartcs/ansible-handson/tree/2021.09.16

SEIKO

SEIKO SOLUTIONS INC.