

The logo for SmartCS, featuring the text "SmartCS" in a bold, blue, sans-serif font.

SmartCS × Ansible連携

説明資料

コンソールサーバー SmartCS と Ansible の連携を実現する  
SmartCS modules for Ansible 、ベンダーモジュールとの連携  
について説明した資料となります。

## ◆内容

- SmartCSの概要
- SmartCSとAnsibleの連携方法
  - SmartCS modules for Ansible
  - ベンダーモジュールとの連携
- 関連資料



ANSIBLE

## SmartCSの役割

- ・ コンソールポートとは
- ・ SmartCSの役割について



## ■ コンソールポートの役割

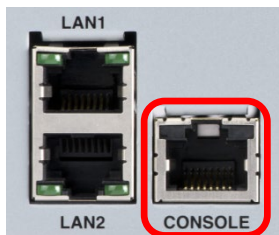
IP通信ではなく、シリアル通信でオペレーションをするためのインターフェース

### ・ 初期設定

IP設定、ユーザ作成、SSHの有効化 などの初期設定

### ・ 緊急時のオペレーション

LANポート障害、ネットワーク障害 などの影響で、  
装置へIPアクセスできなくなった際の最後のアクセス手段



RJ45



DB9

## ■ SmartCSの役割

コンソールポートを集約し、リモートでアクセスできるようにする装置

### ・ リモートアクセス

IPアクセスできない状態の装置へ、リモートでアクセス可能

### ・ オペレーション範囲の拡大

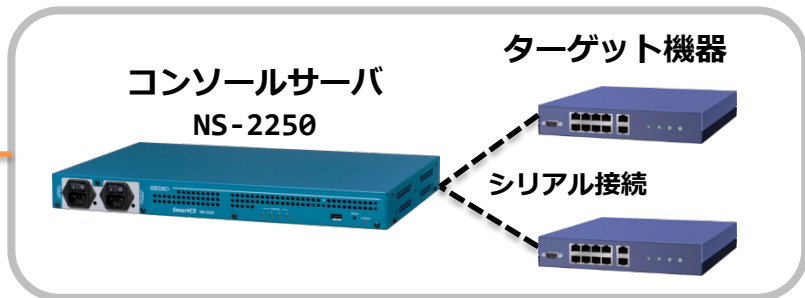
リモートからは実施しづらい作業を、安全に実行可能

ACL/ルーティングなどの設定変更、ファームウェアアップデート など

## オペレーションセンター



## 遠隔地のDCなど



## SmartCSとAnsibleの連携

- ・ 必要となるもの
- ・ 連携してできること
- ・ 連携方法
  - SmartCS modules for Ansible
  - ベンダーモジュールとの連携



## ■ 必要となるもの

### 構成



Ansibleホスト  
・ Ansibleをインストール  
するホストOS

Ansible



SmartCS用Ansibleモジュール  
(SmartCS modules for Ansible)



シリアル接続



SmartCS  
NS-2250シリーズ  
・ NS-2250-16/16D  
・ NS-2250-32/32D  
・ NS-2250-48/48D

ターゲット装置

※NS-2250と  
接続可能な装置

## ■ SmartCS Modules for Ansible の提供について

① Ansible Galaxy サイトからダウンロード、インストールが可能です。

<https://galaxy.ansible.com/seiko/smartcs>

The screenshot shows the Ansible Galaxy page for the 'seiko/smartcs' collection. The breadcrumb path is 'Community Authors > seiko > smartcs'. The collection name is 'SmartCS smartcs', described as 'Ansible Network Collection for Seiko SmartCS devices'. It has 21 downloads and includes buttons for 'Follow Collection', 'Issue Tracker', 'Repo', 'Website', and 'Docs Site'. Navigation tabs include 'Details', 'Read Me', and 'Content'. The 'Info' section shows the installation command: `$ ansible-galaxy collection install seiko.smartcs`. A note states: 'NOTE: Installing collections with ansible-galaxy is only supported in ansible 2.9+'. There is a 'Download tarball' link and an 'Install Version' dropdown set to '1.3.0 released a day ago (latest)'. Tags include 'seiko', 'smartcs', and 'networking'. The 'Content Score' section shows a 'Community Score' of 'No Surv...' (0/5) based on 0 surveys. A 'Tell us about this collection' section contains five survey questions with 'Y' (Yes) and 'N' (No) buttons: 'Quality of docs?', 'Ease of use?', 'Does what it promises?', 'Works without change?', and 'Ready for production?'.



## ■ SmartCS Modules for Ansible の提供について

②Ansible Automation Hubサイトからダウンロード、インストールが可能です。

<https://www.ansible.com/products/automation-hub>

The screenshot shows the Ansible Automation Hub interface for the 'smartcs' collection. At the top, there is a breadcrumb trail: 'Partners > seiko > smartcs'. Below this, the collection name 'smartcs' is displayed with a version dropdown menu set to 'v1.4.0'. A navigation bar includes links for 'Details', 'Documentation', 'Contents', 'Import log', 'Docs site', 'Website', 'Issue tracker', and 'Repo'. The 'Details' section is active, showing the title 'Info' and the description 'Ansible Network Collection for Seiko SmartCS devices'. There are three tags: 'networking', 'seiko', and 'smartcs'. Under the 'License' section, there is a 'License' label. The 'Installation' section contains a terminal command: 'ansible-galaxy collection install seiko.smartcs' with a copy icon. Below the command is a note: 'Note: Installing collections with ansible-galaxy is only supported in ansible 2.9+'. A 'Download tarball' link with a download icon is also present. At the bottom, the 'Install Version' section shows '1.4.0 released 2 months ago (latest)' with a dropdown arrow.

## ■ SmartCS Modules for Ansible の提供について

③弊社ホームページより提供しています。

以下のURLよりお申込み下さい。

[https://www.seiko-sol.co.jp/products/console-server/console-server\\_download/](https://www.seiko-sol.co.jp/products/console-server/console-server_download/)

## ■ 提供内容

項目	内容
NS-2250システム	NS-2250 最新 F/W
<b>SmartCS用 Ansibleモジュール</b>	<b>SmartCS modules for Ansible</b>
ドキュメント	NS-2250 リリースノート
	NS-2250 取扱説明書
	NS-2250 コマンドリファレンス
	<b>NS-2250 Ansible運用ガイド</b>
	NS-2250 バージョンアップ手順書

# SmartCS Modules for Ansible

## ■ SmartCS Modules for Ansible 動作環境について

<v1.0~v1.2>弊社独自のパッケージで提供しています。

SmartCS modules for Ansible		コントロールノード環境		マネージドノード環境 SmartCS システムソフトウェア Ver	
Release	version	ansible (ansible-base)	Python	NS-2250 series	NS-2240 series
2019.4	v1.0	2.7.7	2.7以降 / 3.6以降	V2.0以降	未サポート
2019.10 2021.1	v1.1 v1.1.1	2.8.4		V2.1以降	
2021.1	v1.2	2.9.15	3.6.8		

※NS-2250のSWとAnsible用モジュールは各バージョン毎に対応した組み合わせで動作します。

# SmartCS Modules for Ansible

## ■ SmartCS Modules for Ansible 動作環境について

<v1.3.0～>Ansible Collections用のパッケージで提供しています。

SmartCS modules for Ansible		コントロールノード環境	マネージドノード環境 SmartCS システムソフトウェア Ver	
Release	Version	ansible	NS-2250 series	NS-2240 series
2021.4	v1.3.0	2.10.x (>=2.10, < 2.11)	V2.1以降	未サポート
2021.7	v1.4.0	ansible 2.9.22～ ansible-base 2.10.x ansible-core 2.11.x (>=2.9.22, < 2.12)		
2021.9	v1.4.1	ansible 2.9.10～ ansible-base 2.10.x ansible-core 2.11.x (>=2.9.10, < 2.12)		

※NS-2250のSWとAnsible用モジュールは各バージョン毎に対応した組み合わせで動作します。

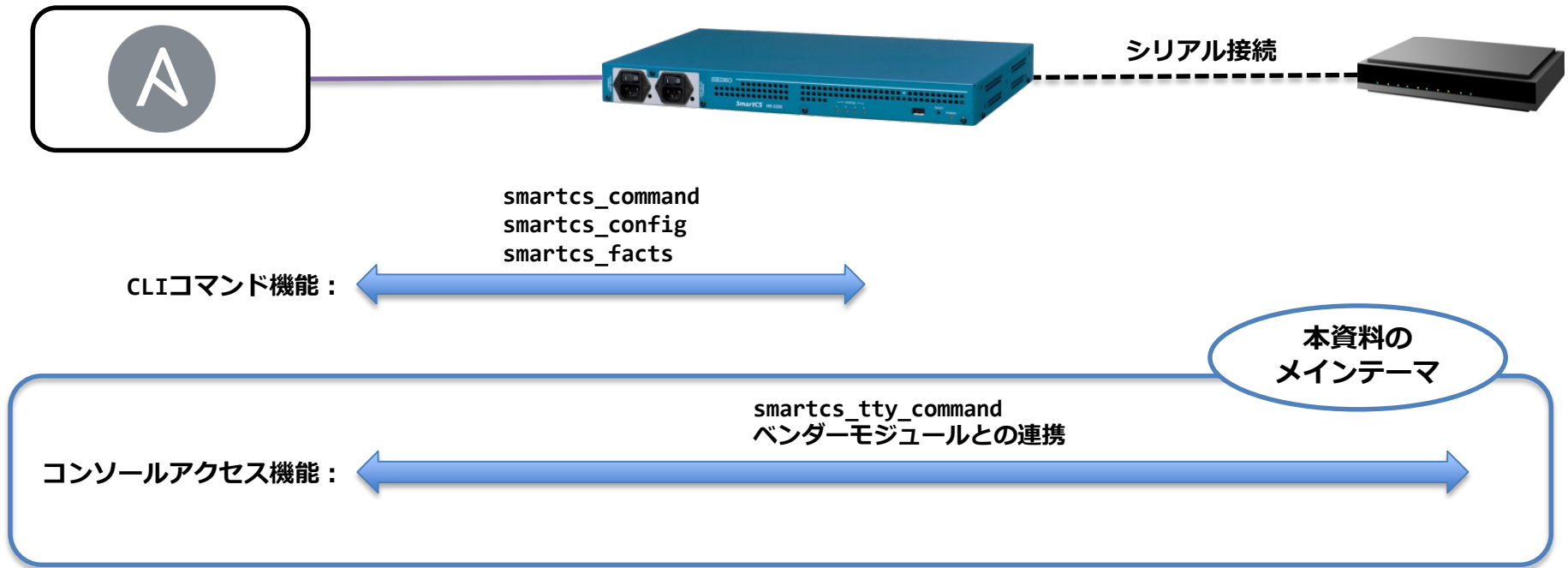
※v1.3.0以降のモジュールはAnsible Galaxyサイト (<https://galaxy.ansible.com/seiko/smartcs>)、v1.4.0以降のモジュールはAnsible Automation Hubサイト (<https://www.ansible.com/products/automation-hub>) で公開しています。

## ■ 連携してできること

SmartCSとAnsibleの連携により、以下の機能が利用可能となります。

機能名	モジュール	内容
CLIコマンド機能	smartcs_command	SmartCSのCLIコマンドを、Ansible経由で実行します。
	smartcs_config	
	smartcs_facts	
コンソールアクセス機能	smartcs_tty_command	SmartCSのシリアルポートに接続されているターゲット装置(NW機器)のコンソールに対して、文字列を送信します。 ※本機能では、SmartCSのttyマネージ機能を使用します。
	ベンダーモジュール	ベンダーモジュールと連携して、SmartCSのシリアルポートに接続されているターゲット装置(NW機器)のコンソールに対して、文字列を送信する機能となります。 ※本機能では、SmartCSのSSHトランスペアレント接続機能(sshxpt機能)を使用します。

## ■ 各機能のオペレーション範囲



## ■ Ansibleとの連携方法は 2パターン用意

	smartcs_tty_command	ベンダーモジュールと連携
特徴	<ul style="list-style-type: none"><li>SmartCS用の独自モジュール</li><li>TeraTermマクロのように、CLIの入出力を定義してPlaybookを作成</li></ul>	<ul style="list-style-type: none"><li>ベンダーモジュールをSmartCS経由(コンソール経由)で実行可能</li></ul>
利点	<ul style="list-style-type: none"><li>モジュールがないターゲットの管理</li><li>既存モジュールでは実現が難しい作業(reboot/verup など)に対応する場合</li><li>1つのPlaybookで全ての処理を行いたい場合</li></ul>	<ul style="list-style-type: none"><li><b>Playbookの再利用が可能</b> ※vars変数のみで切替</li><li>冪等性が担保される</li></ul>
注意	<ul style="list-style-type: none"><li>冪等性の担保がない</li><li>Playbook作成に入出力情報が必要</li></ul>	<ul style="list-style-type: none"><li>network_cli をサポートしているモジュールのみ連携可能</li></ul>

## SmartCS modules for Ansible

- ・モジュールの解説





## ■ SmartCS modules for Ansible で提供するモジュール

## 【CLIコマンド機能】

モジュール	内容
smartcs_command	SmartCSに状態表示コマンド、メンテナンスコマンドを実行し、その実行結果を取得します。 このモジュールは、設定コマンドの実行をサポートしません。 SmartCSの設定を行う場合は、smartcs_configモジュールを使用して下さい。
smartcs_config	SmartCSに設定コマンドを実行します。 設定投入時の指定(lines、src)は、show config running で出力される内容で指定して下さい。
smartcs_facts	SmartCSから装置情報を収集します。 指定可能なオプションは、all、defaule、tty、config となります。

## 【コンソールアクセス機能】

モジュール	内容
smartcs_tty_command	SmartCSのシリアルポートに接続されている監視対象機器のコンソールに対して、 指定された文字列を送信し、コンソールの入出力結果を取得します。

## ■ smartcs\_tty\_command モジュールのポリシー

### コンソールの初期状態について

監視対象機器のコンソール状態について、`smartcs_tty_command`モジュールでは管理、制御を行いません。最後に実行したコマンドによって、監視対象機器のコンソールは、

- ・ログインプロンプト状態
- ・一般ユーザシェル状態
- ・管理者ユーザシェル状態
- ・設定投入用シェル状態

などと、様々な状態が想定されますが、監視対象機器のコンソール状態を考慮してPlaybookを作成して下さい。

### コンソールの入出力結果について

監視対象機器のコンソールで実行されたCLIコマンドの結果について、`smartcs_tty_command`はその実行結果のエラー有無等を自動で判別しません。コンソール上で実行されたCLIコマンドの結果に応じて`ansible`コマンドの実行結果(ok/failed)を制御したい場合は、

- ・`error_recvchar_regex`オプション
- ・`error_detect_on_module`オプション

をご利用して制御を行って下さい。

## ■ smartcs\_tty\_command で使用するオプション

オプション名	設定値	内容
tty	1~48	文字列を送信するSmartCSのシリアルポート番号です。 1-10の様にリスト形式でも指定可能です。
cmd_timeout	1~7200	文字列を送信してから、recvcharの受信待ちがタイムアウトするまでの時間です。
nl	<u>cr</u> / lf/ <u>crlf</u>	送信文字列として「 <u>__NL__</u> 」を指定した際に送信する改行コードです。
sendchar(src)		指定したttyに送信する文字列のリストです。リストの上から順番に送信します。 改行コードや制御文字も送信可能です。  【オプション】 <u>__WAIT__</u> :sec 上述のcmd_timeoutを送信文字列毎に指定するオプションです。  【オプション】 <u>__NOWAIT__</u> recvcharで指定した文字列を待たずに、すぐに次の文字列を送信します。  【オプション】 <u>__NOWAIT__</u> :sec recvcharで指定した文字列を待たずに、指定した時間経過後に次の文字列を送信します。
recvchar (recvchar_regex)		文字列を送信後、受信を期待する文字列(プロンプト等)のリストです。 リスト内のいずれかを受信すると、次の文字列を送信します。 期待する文字列は正規表現での記述も可能です。

## ■ smartcs\_tty\_command で使用するオプション

オプション名	設定値	内容
error_detect_on_sendchar	<u>cancel</u> / exec	文字列を送信後、エラーが発生した場合に、次の文字列を送信するかどうかを指定します。
error_detect_on_module	<u>ok</u> / failed	文字列を送信後、エラーが発生した場合に、ansibleコマンド(ansible-playbookコマンド)の実行結果をokとするかfailedとするかを指定します。
error_recvchar_regex		文字列を送信後、エラーと判定したい受信文字列を正規表現で記述したリストです。
ttycmd_debug	<u>off</u> / on / detail	文字列送受信処理が終了した後、デバッグ情報を表示します。

## ■ smartcs\_tty\_command で使用するオプション

オプション名	設定値	内容
initial_prompt		initial_prompt_check_cmd送信後に受信を期待する文字列です。 (「Login:」など)
initial_prompt_check_cmd		文字列送信の前にコンソールの状態を確認するためのコマンドを指定します。 (改行送信など)
initial_prompt_check_cmd_timeout	1~30	initial_prompt_check_cmd送信後に受信文字列をチェックするまでの時間を指定します。
escape_cmd		initial_promptを受信できなかった場合に送信するコマンドを指定します。 (「exit」など)
escape_cmd_timeout	1~30	escape_cmd送信後に受信文字列をチェックするまでの時間を指定します。
escape_cmd_retry	0~8	escape_cmd送信後にinitial_promptを受信できなかった場合に、initial_prompt_check_cmdの送信リトライ回数を指定します。

## ■ smartcs\_tty\_command で使用するオプション

オプション名	設定値	内容
custom_response	boolean値	stdout、stdout_linesに加えて、sendcharオプションで指定した文字列ごとに、送信文字列(execute_command)と受信文字列(response)が分かれたフォーマットで出力するかどうかを指定します。
custom_response_delete_nl	boolean値	custom_responseの出力内容について、改行のみの行を削除するかどうかを指定します。
custom_response_delete_lastline	boolean値	custom_responseの出力内容について、responseの最終行を削除するかどうかを指定します。 recvcharオプションで指定した文字列のうち、受信した文字列(主にターゲット装置のプロンプト)がresponseに含まれないようにすることが可能です。

## ■ Playbook例

```
---
- 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> "
          : 省略
        sendchar:
          - __NL__
          - cisco
          - cisco
          - enable
          : 省略

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

## ■ recvchar (recvchar\_regex)

- コマンド送信後に期待する文字列（プロンプト等）を複数します。
- 指定したいずれかの文字列を受信したら次の sendchar 文字列を送信します。

## ■ sendchar

- 指定した tty に送信する文字を設定します。
- リストの上から順番に送信します。

• **ansible\_command\_timeout**

→ コンソール経由でコマンドを実行する為、通常のモジュールよりも処理時間が掛かります。その為、タイムアウト値を延長する必要があります。（default: 10s）

• **ansible\_connection**

→ **network\_cli** を指定

• **ansible\_network\_os**

→ **smartcs** を指定

• **ansible\_user** , **ansible\_password**

→ SmartCSにログインする為の拡張ユーザのログイン情報を指定します。

## ■ Playbook実行結果

名前	説明	契機	タイプ
stdout	コマンドの実行結果	コマンドの実行に成功した場合	リスト
stdout_lines	コマンド実行結果を送信文字列毎に分割したリスト		リスト

### stdout出力例

```
"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出力例

```
"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#"
  ]
],
```



## ■ Playbook実行結果

名前	説明	契機	タイプ
stdout_lines_custom	コンソールの送受信文字列について、送信文字列 (execute_command)、受信文字列 (response) を区別した形式のリスト。	custom_response設定が有効、かつコマンドの実行に成功した場合	リスト

## オプション設定値

- custom\_response : **on**  
⇒ stdout\_lines\_customでの出力有効
- custom\_response\_delete\_nl : **on**  
⇒ コマンド実行結果の行間を削除
- custom\_response\_delete\_lastline : **off**  
⇒ 最終行(プロンプト等)は削除しない

## 出力例

```
"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#"  
    ]  
  },  
]
```

## ■ sendchar の送信可能文字について

- 送信可能文字列は recvchar と同様に可視化文字全て。

### - 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 { | } ~
```

- モジュールv1.0では、上記の赤文字記号について sendchar オプションで送信不可。モジュールv1.1では、recvchar と同様に全ての可視化文字を送信可能。
- sendchar で一部記号を指定する場合、はシングルコーテーションやダブルコーテーションで囲う必要があります。' (シングルコーテーション)、" (ダブルコーテーション) など

### - 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 { | } ~
```

## ■ sendchar の送信可能文字について

- sendchar で 制御文字を送信可能。

### 送信可能な制御文字

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
```

- sendchar 同様、各送信オプションにも対応

```
__CTL__:03
__CTL__:03 __WAIT__:30 :送信後、recvchar の返りを指定時間待つ
__CTL__:03 __NOWAIT__:30 :送信後、指定時間待つ(recvchar待たない)
__CTL__:03 __NOWAIT__ :送信後、すぐに 次のsendcharを送信する
```

## ■ sendchar を src で指定

- sendchar で送信する文字列について 外部ファイルを指定可能。

### Playbook

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

### 外部ファイル

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

sendchar\_cisco.txt

## • 注意

- sendchar と src オプションはどちらかのみ指定可能 (排他設定)

## SmartCS と Ansible の連携

## ■ sendchar を src で指定

## ・ src オプションを利用した Playbook 例

## 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 }}'

```

・vars として cs\_parameters (dict型) を指定  
※group\_vars, host\_vars に定義しても可

・dict内のKeyとして、tty, config を用意して  
それぞれパラメータを指定

tty 情報

src 情報  
※sendchar

## 処理フロー

Playbook 1 回の実行で  
tasks の smartcs\_tty\_command の 処理について  
cs\_parameters の値を参照して実行される。  
※recvcharは共通

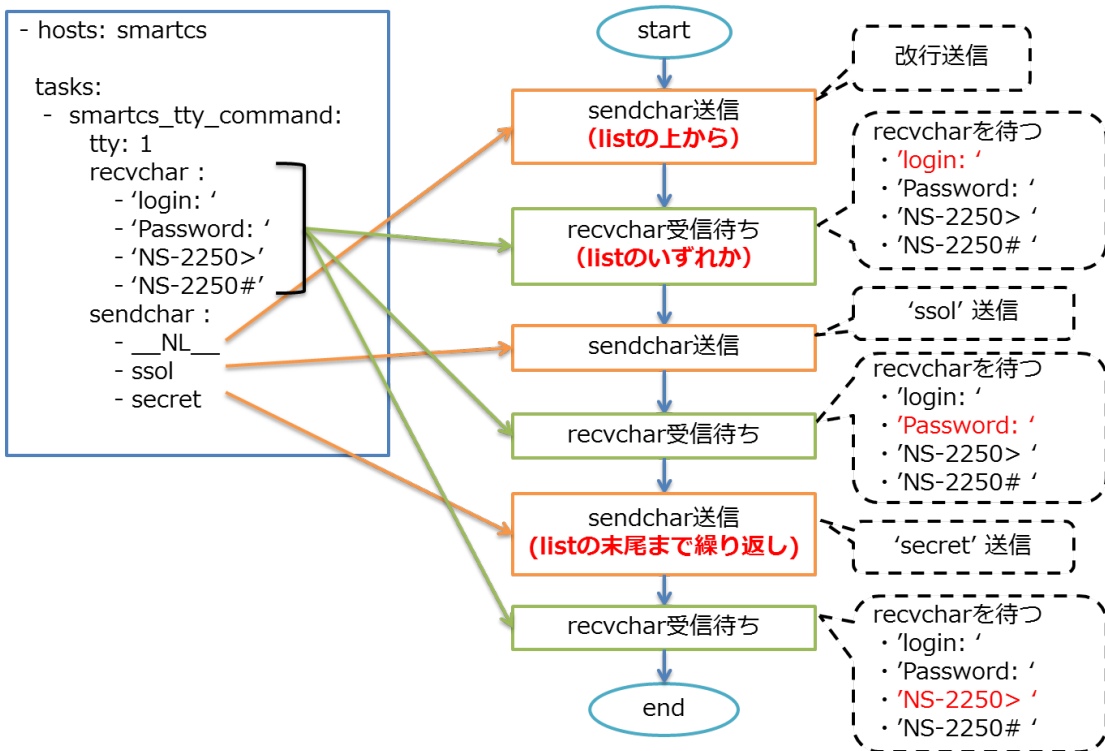
① smartcs\_tty\_command  
- tty: '4'  
- src: './src/config4'



② smartcs\_tty\_command  
- tty: '5'  
- src: './src/config5'

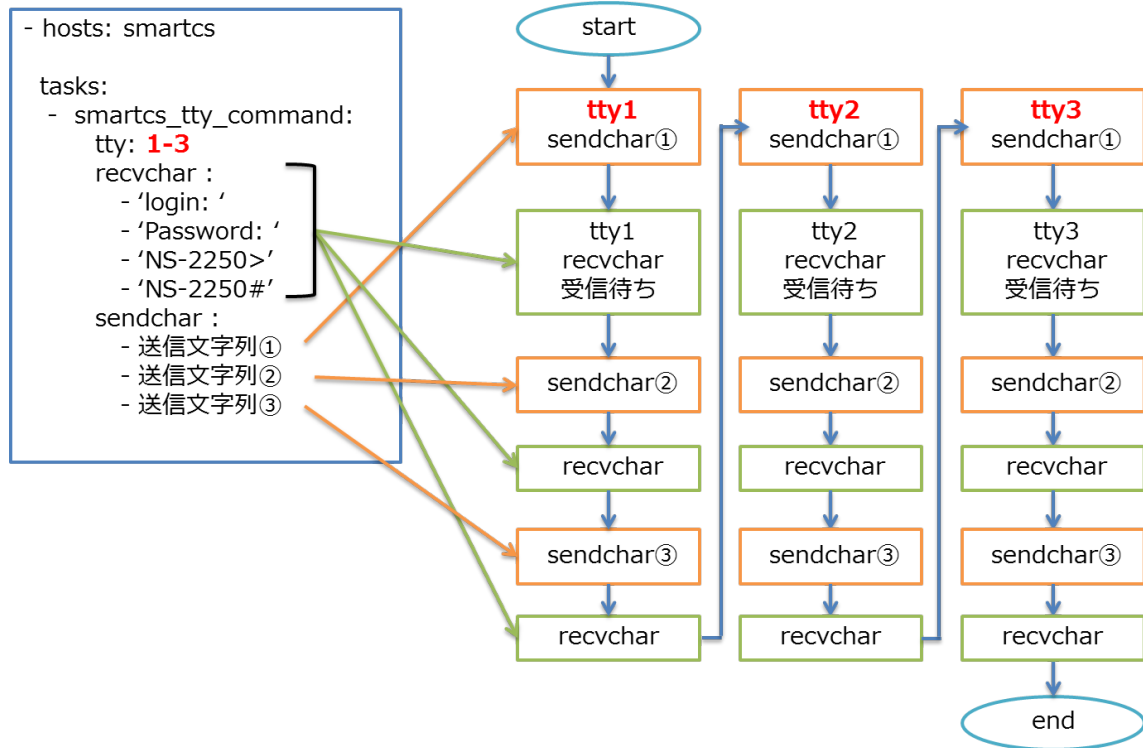
## ■ smartcs\_tty\_command オプション解説

### sendchar / recvchar



## ■ smartcs\_tty\_command オプション解説

ttyを複数指定



## ■ smartcs\_tty\_command オプション解説

### cmd\_timeout

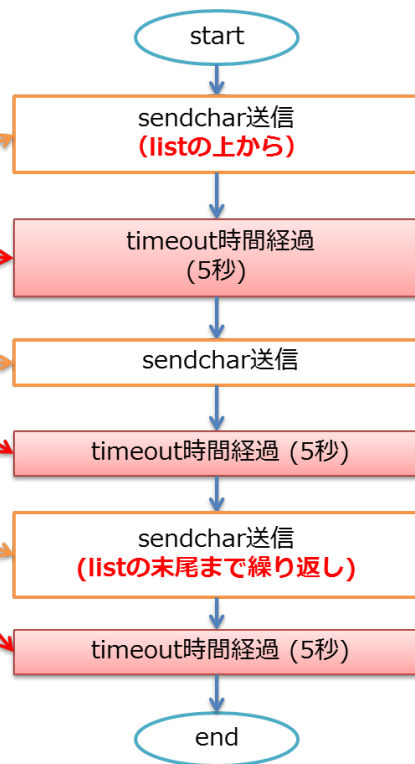
```

- hosts: smartcs

tasks:
- smartcs_tty_command:
  tty: 1
  error_detect_on_sendchar : exec

  cmd_timeout: 5

  sendchar :
  - __NL__
  - ssol
  - secret
  
```





## ■ smartcs\_tty\_command オプション解説

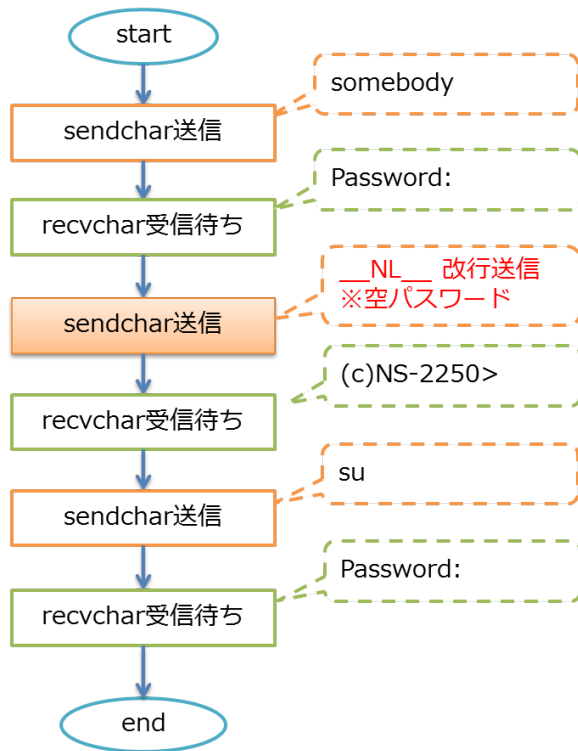
### sendcharのオプション指定 `__NL__` の動作

```

- hosts: smartcs

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

改行だけを送信したい場合に使用



## ■ smartcs\_tty\_command オプション解説

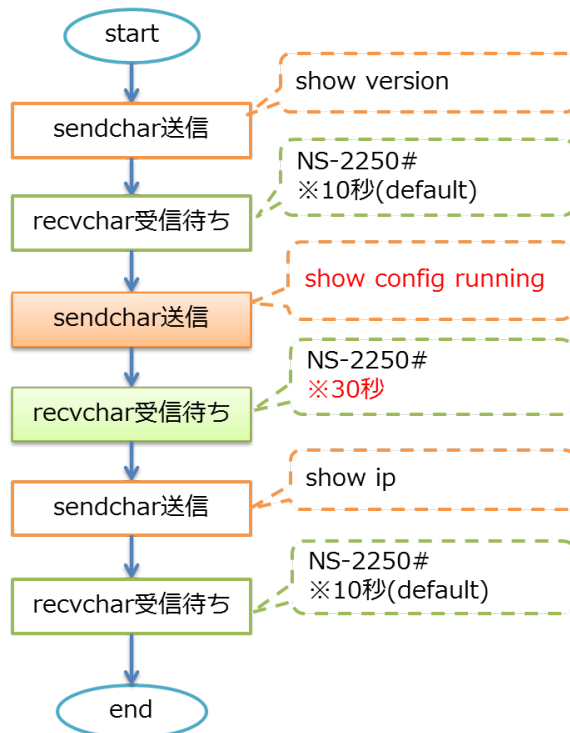
sendcharのオプション指定 `__WAIT__:Xsec` の動作

```

- hosts: smartcs

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

特定のsendcharを送信後、  
タイムアウト時間を変更したい時に使用する事ができます。  
※特定のコマンドのみ実行時間が長い場合など



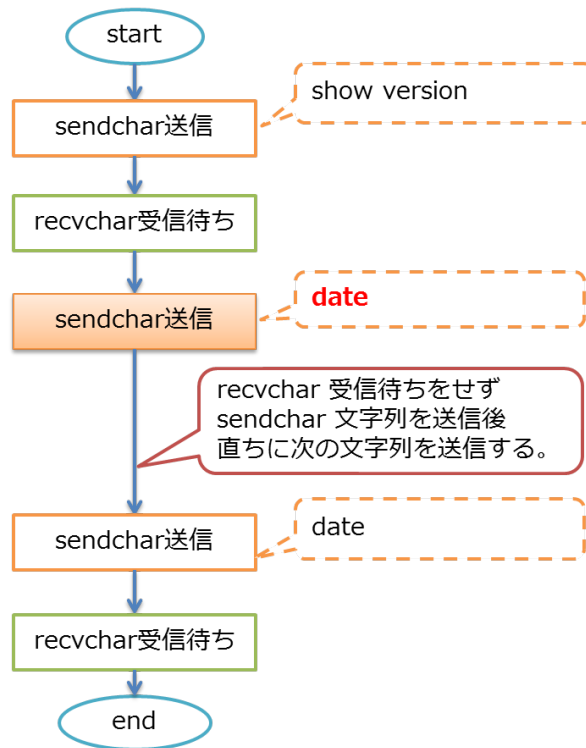
## ■ smartcs\_tty\_command オプション解説

sendcharのオプション指定 `__NOWAIT__` の動作

```
- hosts: smartcs

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

sendchar送信後、rcvcharを待たずに連続して文字列を送信したい場合などに使うことができます。



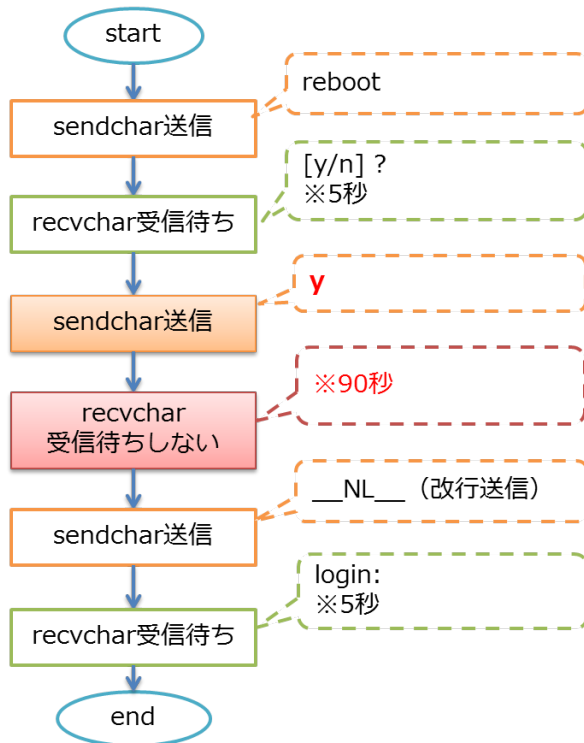
## ■ smartcs\_tty\_command オプション解説

sendcharのオプション指定 `__NOWAIT__`:Xsec の動作

```

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

特定のsendcharを送信後、  
設定したrcvcharを待たない時間を設定する事ができます。  
※reboot時など、本来意図しない部分で  
rcvcharを受信してしまう場合など。



## ■ smartcs\_tty\_command オプション解説

sendcharで設定した文字列を送信する場合、文字列の送信がエラーとなる場合があります。

sendchar送信後に発生するエラー要因		出力
recvcharをタイムアウト時間までに受信できなかった		Error:: Timeout.
対象のttyに接続できない	接続許可設定が無い為、接続できない	Error:: Not allowed.
	排他制御により接続できない	Error:: Session limit over.
	tty管理用デーモンに接続ができない	Error:: Connection closed.
	error_recvchar_regexで設定した文字列を検出した	Error:: Matched “xxxxx”.
error_detect_on_sendchar設定が cancel の時に、次のsendcharを送信しない		Error:: After error.

### ※接続許可が無い場合について

拡張ユーザグループのユーザに適切な権限が無いか、ttyマネージ機能が有効となっていない場合、エラーとなります。

### ※排他処理について

Ansible経由のアクセス（ttyマネージ機能によるアクセス）と、従来のポートユーザによるアクセスは、同時に行う事が出来ません。先に接続した方が優先となります。

### ※error\_recvchar\_regex について

設定していない場合は、有効となりません。

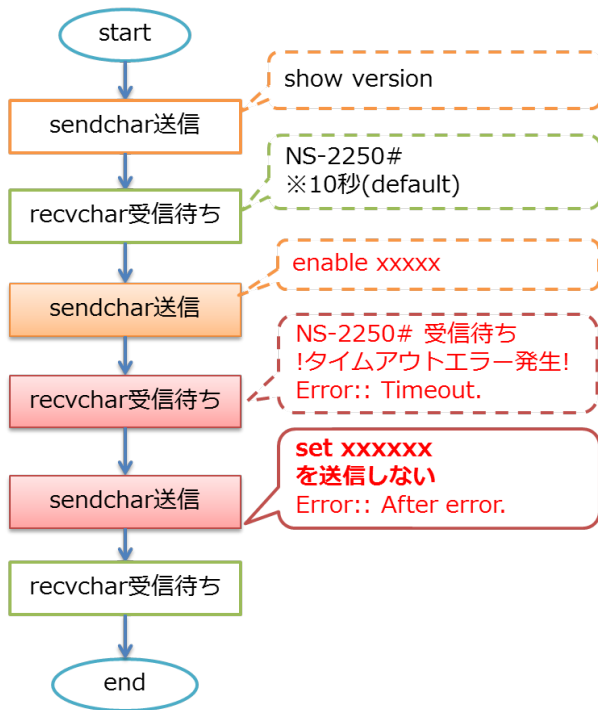
設定している場合、送受信文字列に指定した文字列が含まれている場合、エラーと判定します。

## ■ smartcs\_tty\_command オプション解説

error\_detect\_on\_sendchar オプション cancel 設定時 の動作

```
- hosts: smartcs

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



エラーが発生した場合、エラー発生後のsendcharを送信しない場合に設定します。  
※誤動作防止など

## ■ smartcs\_tty\_command オプション解説

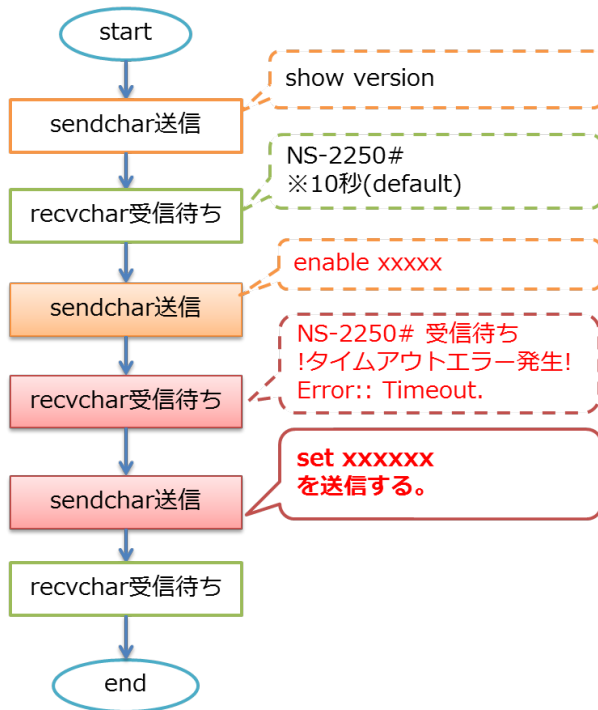
error\_detect\_on\_sendchar オプション exec 設定時 の動作

```

- 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
  
```

エラーが発生した場合でも、  
sendcharを送信する場合に設定します。



## ■ smartcs\_tty\_command オプション解説

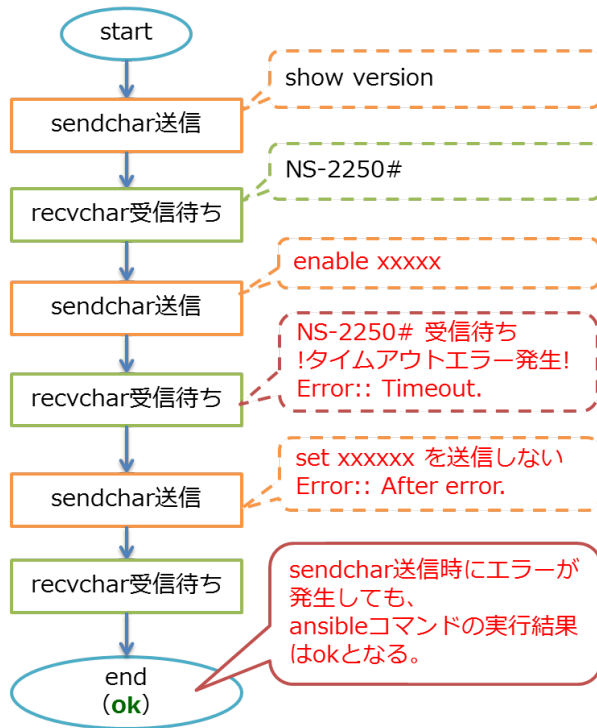
error\_detect\_on\_module オプション ok 設定時 の動作

```

- hosts: smartcs

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

エラーが発生した場合でも、  
ansibleコマンドはfailedとせず、okとする場合に設定します。  
※判定可能なエラーは、  
「sendchar送信後に発生するエラー要因」  
の内容に限ります。





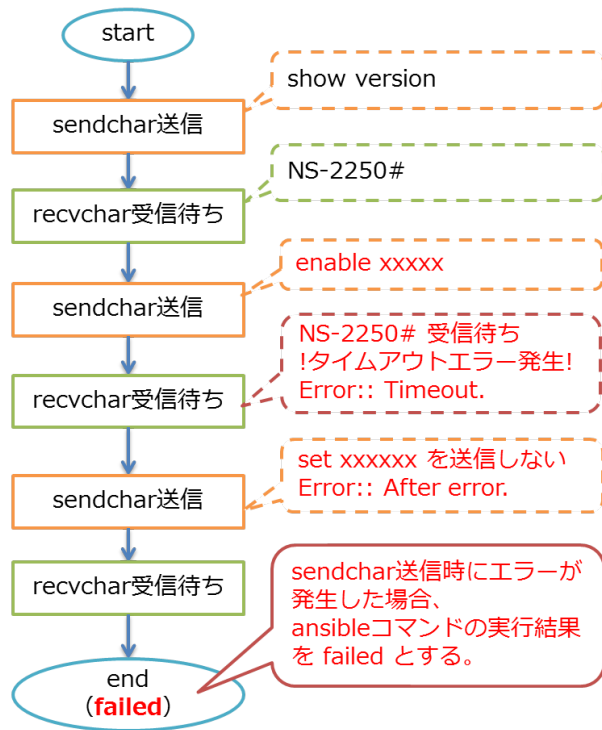
## ■ smartcs\_tty\_command オプション解説

### error\_detect\_on\_module オプション failed 設定時 の動作

```

- hosts: smartcs

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



エラーが発生した場合、ansibleコマンドをfailedとします。  
 ※判定可能なエラーは、  
 「sendchar送信後に発生するエラー要因」の内容に限ります。  
 ※AnsibleTowerやAWXとの連携時など

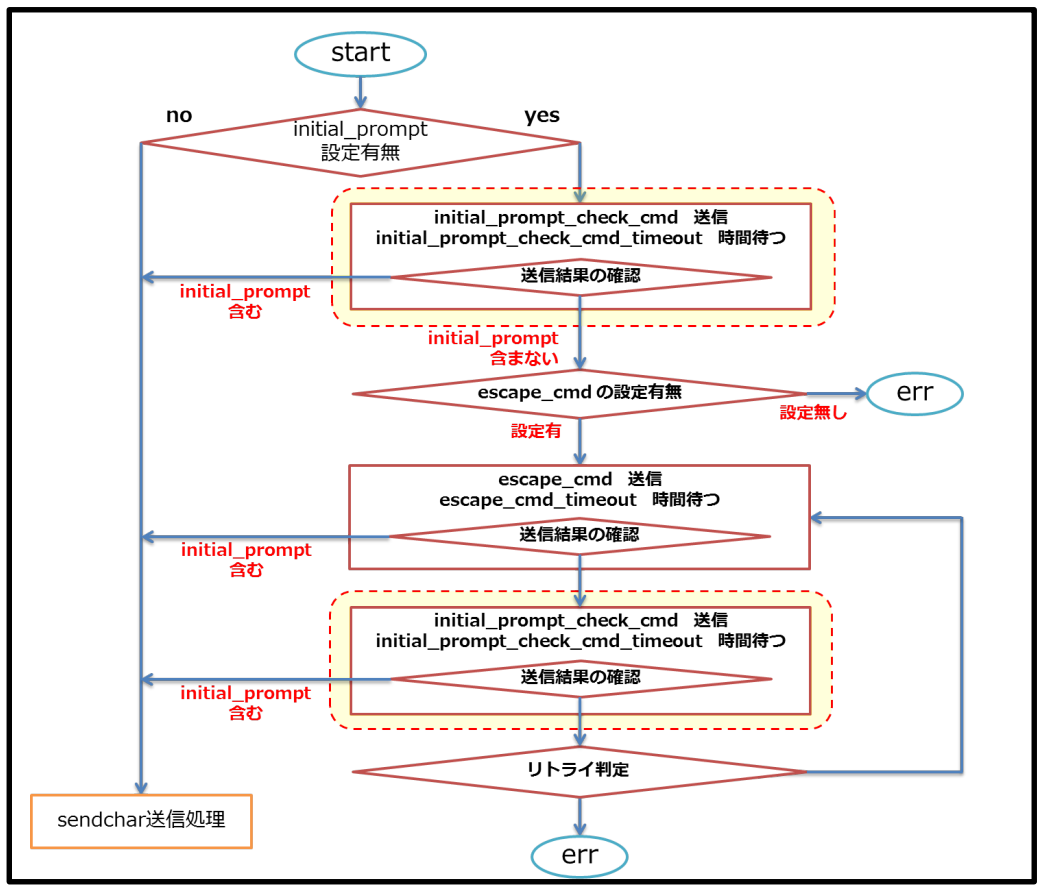
## ■ sendchar の送信前にコンソールの状態をチェックする機能

- sendchar 送信前にコンソールの状態が期待する状態かを確認します。

### 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
```

オプション名	内容
initial_prompt	期待する文字列を設定します。 正規表現による設定も可能です。
initial_prompt_check_cmd	sendchar送信前にコンソールの状態を確認する為のコマンドを設定します。 デフォルト値は改行(__NL__)です。
initial_prompt_check_cmd_timeout	チェック用コマンド送信後のタイムアウト値を設定します。 ( default 3s )
escape_cmd	期待する文字列が出力されなかった場合に送信するコマンドを設定します。 例 (exit , logout 等)
escape_cmd_timeout	escape_cmd のタイムアウト値を設定します。 ( default 3s )
escape_cmd_retry	リトライ回数を設定します。 ( default 3 )



## ■ smartcs\_tty\_command の返り値の拡張

- stdout, stdout\_lines 以外にコンソールの入出力を分かりやすいフォーマットで出力する返り値 (stdout\_lines\_custom) を追加するオプションを用意。

### 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
```

オプション名	内容
custom_response	stdout, stdout_lines に加えて sendchar , recvchar を区別できるフォーマットの返り値を返します。 (stdout_lines_custom) sendchar 毎に execute_command, response を分けて出力します。
custom_response_delete_nl	custom_response の出力内容について、改行だけの行を削除します。
custom_response_delete_lastline	custom_response の出力内容について、responseの最後の1行を削除します。 ※CLIコマンド実行後のプロンプトを表示させない事を目的としています。 ※show系コマンドの実行結果のみを出力

## ■ smartcs\_tty\_command の返り値の拡張

### 設定例

- custom\_response : off
- custom\_response\_delete\_nl : off
- custom\_response\_delete\_lastline : off

### 表示例

<従来のstdout\_linesと同様>

```

"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#"
  ],
  [
    "date",
    "",
    "Thu Sep 26 15:18:54 JST 2019",
    "",
    "(c)NS-2250#"
  ],
],

```

sendchar送信

recvchar受信

recvchar受信

sendchar送信

recvchar受信

コマンド (sendchar) 実行結果

コマンド (sendchar) 実行結果

## ■ smartcs\_tty\_command の返り値の拡張

### 設定例

- custom\_response : **on**
- custom\_response\_delete\_nl : off
- custom\_response\_delete\_lastline : off

### 表示例

<v1.1で拡張する stdout\_lines\_custom>

```

" 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#"
    ]
  }
]

```

sendchar送信

recvchar受信

recvchar受信

sendchar送信

recvchar受信

コマンド (sendchar) 実行結果

コマンド (sendchar) 実行結果

## ■ smartcs\_tty\_command の返り値の拡張

## 設定例

- custom\_response : **on**
- custom\_response\_delete\_nl : **on**
- custom\_response\_delete\_lastline : off

sendchar 送信後の実行結果から

**空白行を削除**

→コマンド実行結果の視認性と抽出時の手間を考慮  
※Playbook側で行う整形作業のSTEPを減らす

## 表示例

<v1.1で拡張する stdout\_lines\_custom>

```
" 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#"  
    ]  
  }  
]
```

## ■ smartcs\_tty\_command の返り値の拡張

## 設定例

- custom\_response : **on**
- custom\_response\_delete\_nl : **on**
- custom\_response\_delete\_lastline : **on**

sendchar 送信後の実行結果から  
**最終行 (プロンプト) を削除**

→コマンド実行結果のみとなるように整形する。  
※Playbook側で行う整形作業のSTEPを減らす

## 表示例

<v1.1で拡張する stdout\_lines\_custom>

```
" 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",  
    ]  
  }  
]
```



## ■ smartcs\_tty\_command の返り値の拡張

### Playbook 抜粋

```

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",
  ]
}

```

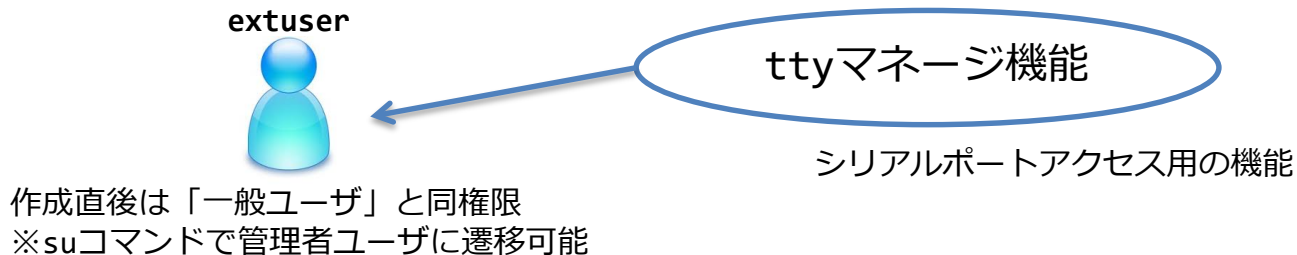
## SmartCS modules for Ansible

- SmartCSに必要な設定



## ■ smartcs\_tty\_commandモジュールを利用するには

- ・ 「拡張ユーザ」グループのユーザ作成
- ・ 「拡張ユーザ」グループのユーザにttyマネージ機能の権限付与
- ・ ttyマネージ機能の有効化 が必要となります。



extuser でログインする事で、  
実行可能なCLIが拡張されたユーザを利用可能に

## ■ smartcs\_tty\_commandモジュールを利用するには

- ・ 「拡張ユーザ」グループのユーザを作成

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

- smartcs\_tty\_command を利用可能な、拡張ユーザグループ(extusr) のユーザを作成。
  - アクセス可能なシリアルポート番号と、パスワードの設定も必要。
  - 設定するユーザ名/パスワードは、Ansibleからアクセスする際の 「ansible\_user」 で指定するユーザ名 と、「ansible\_password」 で指定するパスワード に該当します。
- ・ 作成したユーザに権限を付与

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

- 拡張ユーザグループ(extusr) のユーザに、ttyマネージ機能の権限を付与。
- ・ 機能の有効化

```
enable ttymanage
```

- ttyマネージ機能を有効化。

## ■ 拡張ユーザグループの概要

ユーザグループ	グループ名	権限					
		状態 統計情報表示	装置の設定	装置への Telnet/SSH ログイン	装置への FTP/SFTP ログイン	装置の コンソール ポートへの ログイン	管理対象機器 (シリアルポート) へのアクセス
装置管理ユーザ	root	○	○	○	×	○	×
一般ユーザ	normal	○	×	○	×	○	×
<b>拡張ユーザ</b>	<b>extusr</b>	○	×	○※1	×	×	○※2
ポートユーザ (シリアルポートへのアクセス)	portusr	×	×	×	×	×	○
FTP/SFTP用ユーザ	setup verup log	×	×	×	○	×	×

※1 拡張ユーザは、SmartCSにSSHアクセスでのみログイン可能となります。

※2 拡張ユーザは、SmartCSに接続されている管理対象機器にアクセスする方法として、CLIコマンド(ttyマネージ機能)を使用します。

## ■ その他

- ・ 拡張ユーザグループのユーザの同時接続数：48セッション
- ・ sshd オブジェクトの設定に沿って動作します。(認証方式 / ポート番号 / allowhost / ipfilter)

## ベンダーモジュールとの連携

v2.1以降の  
システムで利用可能

- ・ 連携機能について
- ・ SmartCSに必要な設定



## ■ベンダーモジュールとの連携機能

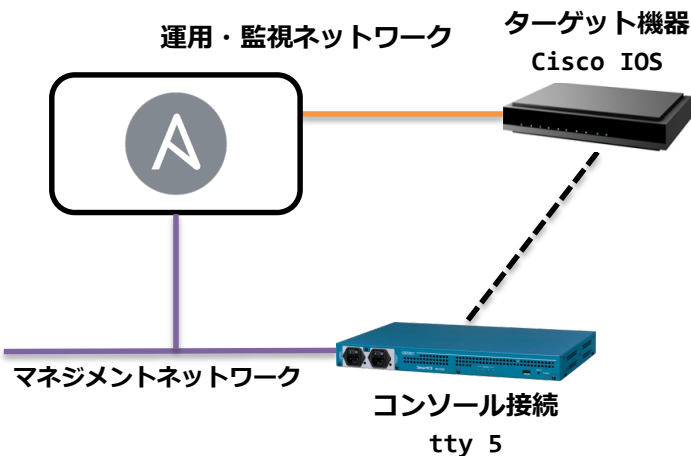
- Cisco機器やArista機器などを制御する為に作成したPlaybookについて、通常はSSH経由で接続して処理を実行しますが、SmartCSのコンソール経由での実行が可能となります。
- Playbookのタスク部分については変更する事なく、再利用が可能です。

### 構成



## ■ベンダーモジュールとの連携イメージ

## 構成



## 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
  
```

接続先はSmartCS

ベンダー  
モジュールを指定コネクションプラグインは  
network\_cliを指定ログインID  
パスワード には  
SmartCSの  
ポートユーザを指定

SSHの接続ポートを変更

OS種別は  
使用する他社ベンダを指定



## ■ 他ベンダーモジュールの実行 (Playbook構成例)



### Playbook ①

Module: smartcs\_tty\_command

ユーザ : 拡張ユーザ  
ポート : SSHポート (22)



### Playbook ②

Module: **他社ベンダーモジュール**

ユーザ : ポートユーザ  
ポート : sshxpt ポート (93xx)



### Playbook ③

Module: smartcs\_tty\_command

ユーザ : 拡張ユーザ  
ポート : SSHポート (22)

SmartCS経由  
装置コンソールへのログイン処理

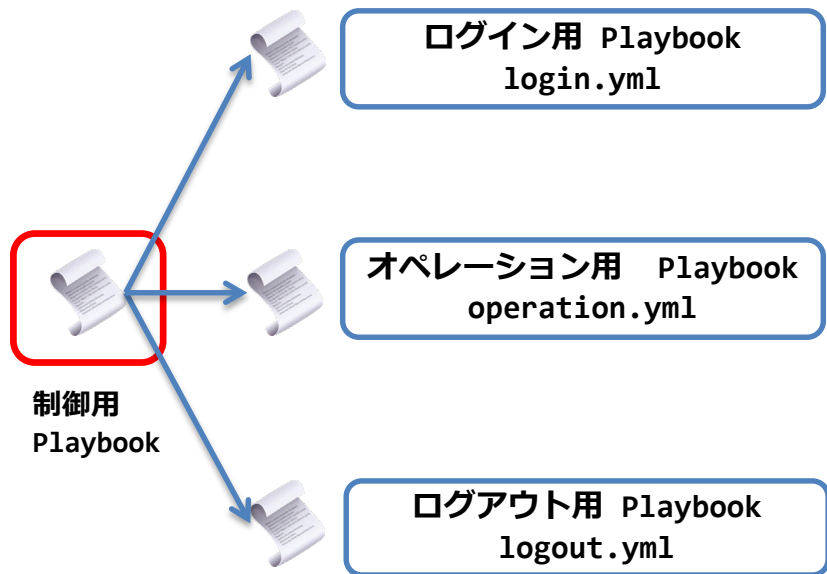


SmartCS経由  
装置の制御 (設定・表示)



SmartCS経由  
装置コンソールからのログアウト処理

## ■ 他ベンダーモジュールの実行 (Playbook構成例)



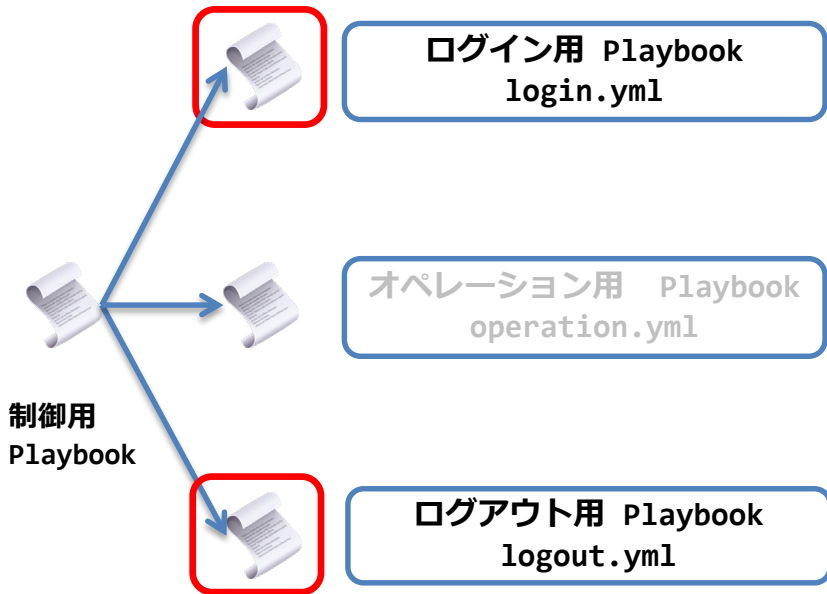
```

---
- 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

```

制御用 Playbook例  
※実際に実行するPlaybook

## ■ 他ベンダーモジュールの実行 (Playbook構成例)



```

---
- 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    ←IOS装置へのログインID
        - secret      ←IOS装置へのログインパスワード
  
```

login.yml

NW機器に応じて  
コンソール経由の  
ログインプロンプトを指定

NW機器の汎用的な  
プロンプト例

装置のコンソールに  
ログインする際の  
ID,パスワードを指定

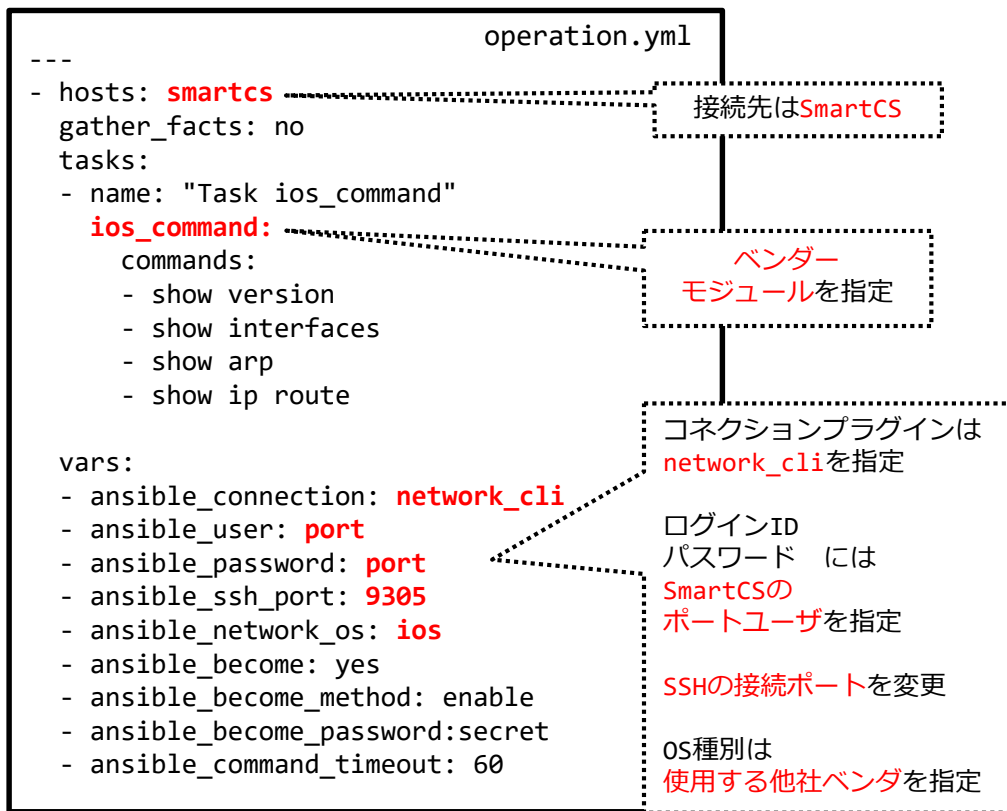
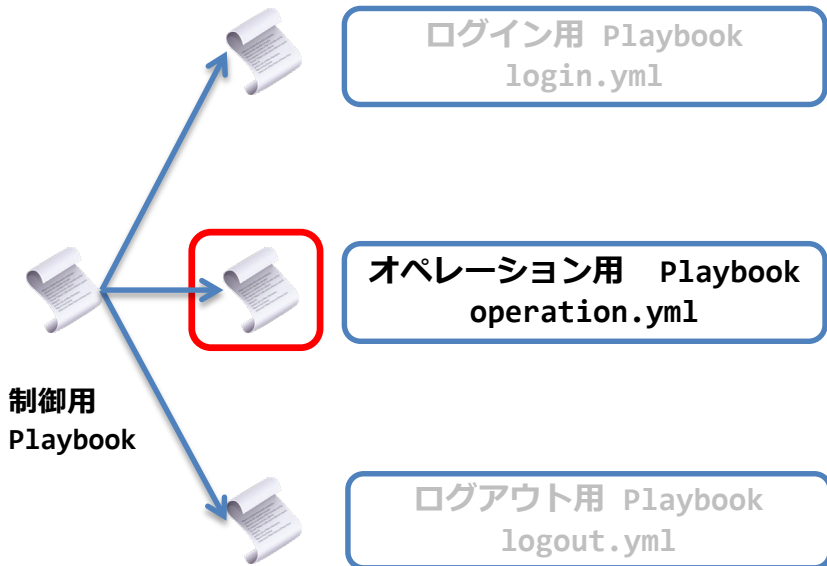
```

---
- 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
  
```

logout.yml

exit 送信回数は  
NW機器に応じて  
複数回指定

## ■ 他ベンダーモジュールの実行 (Playbook構成例)



## ■ 他ベンダーモジュールの実行

### <連携する際に気を付けるポイント>

- ・ 連携する他社ベンダーモジュールについて、`network_cli` をサポートしているものに限る

- ・ SSHで装置にログインしてCLIを実行する処理をコンソール経由で行う内部処理となる為

例

```
vars:  
  - ansible_connection: network_cli
```

- ・ SSH接続時とコンソールアクセス時のプロンプト定義が同じでないと動作しない（`terminal` プラグインの定義）

- ・ 処理速度のケア

- ・ 他社ベンダのモジュールは通常SSH接続して動作するが、本連携ではコンソール経由で動作する事になるその為、処理速度が遅いのでタイムアウト時間の延長が必要。（コマンド実行時間など）

例

```
vars:  
  - ansible_command_timeout: 60
```

## ■ベンダーモジュールと連携するには

- 既存のTCPポートを利用せず、新たなサービスポートを用意

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

- sshxpt オプションを指定する事で新たに 9301~9348 のTCPポートを OPEN し、ポート接続を待ち受ける。
  - 本ポートは、既存のダイレクト / セレクトのサービスポートとは独立に動作する為、既存のサービスに影響を与えません。
  - Ansibleからアクセスする際の「ansible\_port」で指定するポート番号に該当します。
- ポート番号は設定変更が可能です。

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

- 設定範囲 : 1025 ~ 65000
  - デフォルト値 : 9301
- 関連する表示コマンドの対応
    - show portd , show portd tty

## ■ベンダーモジュールと連携するには

- 接続時のアクションを指定（改行コードの送信）

```
set portd tty <ttylist> connted send_n1 { cr | crlf | lf | none }
```

- 接続時に送信する改行コードを指定します。
- デフォルトは none （ sshxpt ポートに接続しても何も送信しない ）

※接続時に改行コードを送信し、プロンプトを出力させる事で

network\_cli プラグインが動作するようにしています。

- 「ポートユーザ」グループのユーザを作成

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

- sshxpt機能 を利用可能な、ポートユーザグループ(portusr) のユーザを作成。
- アクセス可能なシリアルポート番号と、パスワードの設定も必要。
- 設定するユーザ名/パスワードは、Ansibleからベンダーモジュールを利用してアクセスする際に「ansible\_user」で指定するユーザ名 と、「ansible\_password」で指定するパスワード に該当します。

## 参考情報

- WEBINAR
- Ansible Automates Tokyo 2020
- Ansible ハンズオン





## ■過去の講演 & Ansibleハンズオン

### • WEBINAR

Ansibleではじめる“失敗しないITとNWの自動化”  
～ITとNWの自動化におけるSmartCSの重要性～

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

### • Ansible Automates Tokyo 2020

運用自動化を支えるSmartCSの役割 & ユーザ事例紹介

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

### • Ansibleハンズオン

SmartCS × ALAXALA × Ansible ハンズオン

<https://github.com/ssol-smartcs/ansible-handson/blob/2020.01.31/SmartCSxALAXALA/README.md>

SmartCS × IOS × Ansible ハンズオン

<https://github.com/ssol-smartcs/ansible-handson/blob/2021.07.29/SmartCS%C3%97IOS/README.md>

SEIKO

セイコーソリューションズ株式会社