SmartCS × Ansible連携

説明資料

本資料について

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

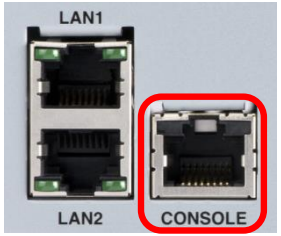
IP通信ではなく、シリアル通信でオペレーションをするためのインターフェース

・**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

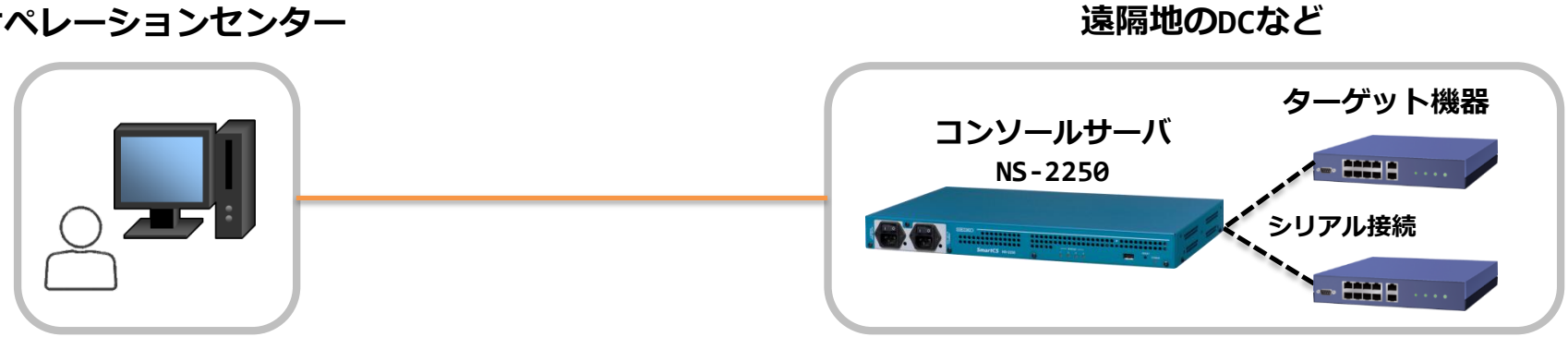
・**リモートアクセス**

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、ファーウェアアップデート　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

ANSIBLE

SmartCS と Ansible の連携

SEIKO

■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)

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

SEIKO

■SmartCS modules for Ansible の提供について

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/

■提供内容

項目	内容
NS-2250システム	NS-2250 最新 F/W
SmartCS用 Ansibleモジュール	**SmartCS modules for Ansible**
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
	NS-2250 Ansible運用ガイド
	NS-2250 バージョンアップ手順書

SmartCS と Ansible の連携

SEIKO

■SmartCS modules for Ansible リリース履歴

NS-2250 SW	SmartCS modules for Ansible	備考	項目	バージョン
v2.0	v1.0	初回リリース	Python	2.7以降 3.6以降
			Ansible	Ansible 2.7.7
v2.1	v1.1	ベンダーモジュールとの連携対応 送信文字と返り値の拡張 など	Python	2.7以降 3.6以降
			Ansible	Ansible 2.8.4

※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 と Ansible の連携

SEIKO

■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。

SmartCS と Ansible の連携

SEIKO

■各機能のオペレーション範囲

smartcs_command
smartcs_config
smartcs_facts

CLIコマンド機能：

本資料の
メインテーマ

smartcs_tty_command
ベンダーモジュールとの連携

コンソールアクセス機能：

SmartCS と Ansible の連携

SEIKO

■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	・SmartCS用の独自モジュール ・TeraTermマクロのように、CLIの入出力を定義してPlaybookを作成	・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
利点	・モジュールがないターゲットの管理 ・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 ・1つのPlaybookで全ての処理を行いたい場合	・**Playbookの再利用が可能** ※vars変数のみで切替 ・冪等性が担保される
注意	・冪等性の担保がない ・Playbook作成に入出力情報が必要	・network_cli をサポートしているモジュールのみ連携可能

SmartCS modules for Ansible

・モジュールの解説

ANSIBLE

SmartCS と Ansible の連携

SEIKO

■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。

SEIKO

■ 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_commnand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。

SEIKO

■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	cr/ lf/ crlf	送信文字列として「__NL__」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】__WAIT__: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】__NOWAIT__ 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】__NOWAIT__: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。

SEIKO

■ 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	cancel / 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	ok / 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	off / 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。

SEIKO

■ 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。

SEIKO

■ 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。

SmartCS と Ansible の連携

SEIKO

■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。

SEIKO

■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#"
        ],

 ],
```

SEIKO

■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#"
            ]
        },
    ]
```

SEIKO

■sendchar の送信可能文字について

・送信可能文字列は recvchar 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 { | } ~
```

SEIKO

■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
```

SEIKO

■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）

SEIKO

■ sendchar を src で指定

・srcオプションを利用したPlaybook例

Playbook

・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

```
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 }}'
```

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

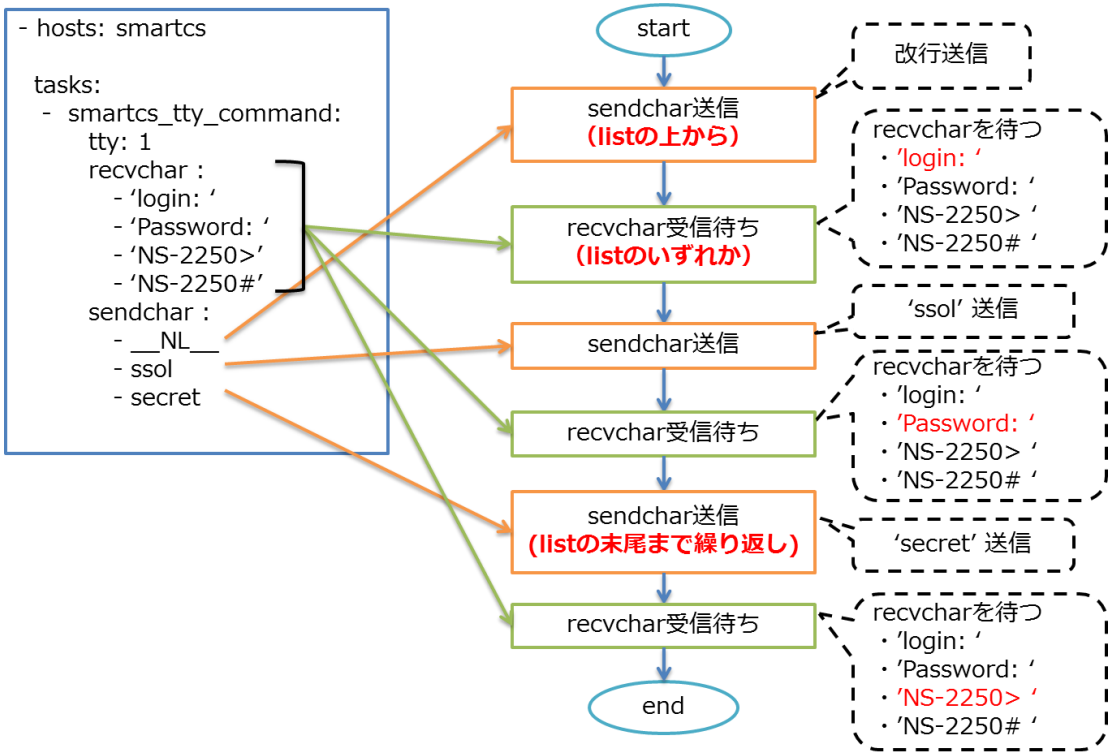
```
①smartcs_tty_command
  - tty: '4'
  - src: './src/config4'
```

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

SEIKO

■ smartcs_tty_command オプション解説

sendchar / recvchar

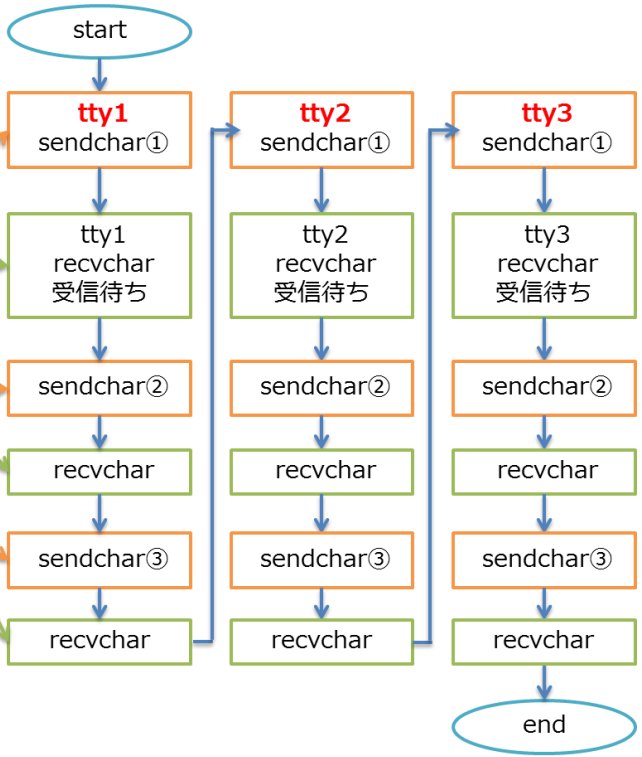


SEIKO

■ smartcs_tty_command オプション解説

ttyを複数指定

```
- hosts: smartcs

  tasks:
   -  smartcs_tty_command:
        tty: 1-3
        recvchar :
          - 'login: '
          - 'Password: '
          - 'NS-2250>'
          - 'NS-2250#'
        sendchar :
          - 送信文字列①
          - 送信文字列②
          - 送信文字列③
```

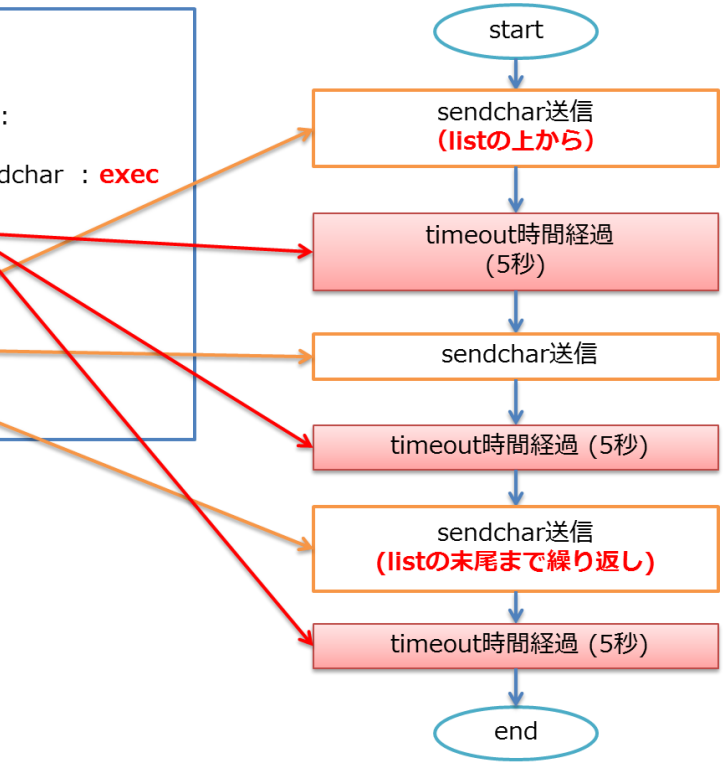


SEIKO

■ 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
```

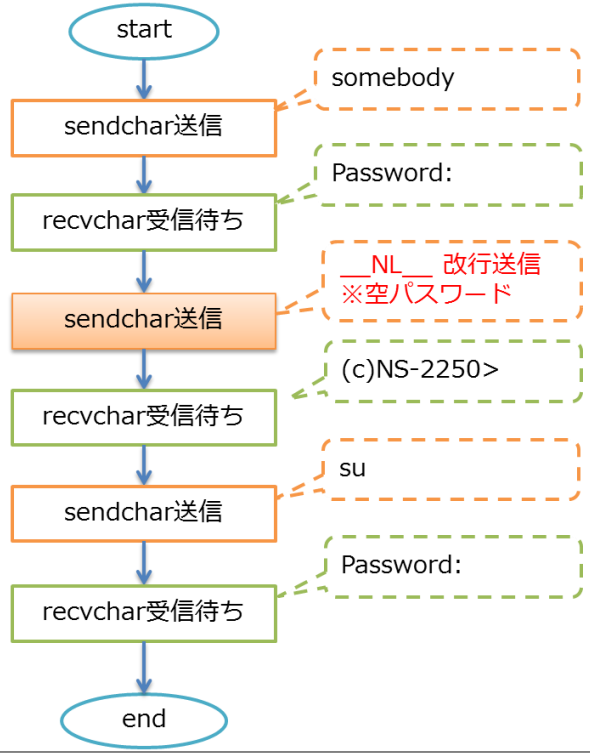


SEIKO

■ 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
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - somebody
         - __NL__
         - su
         - __NL__
         - show version
```



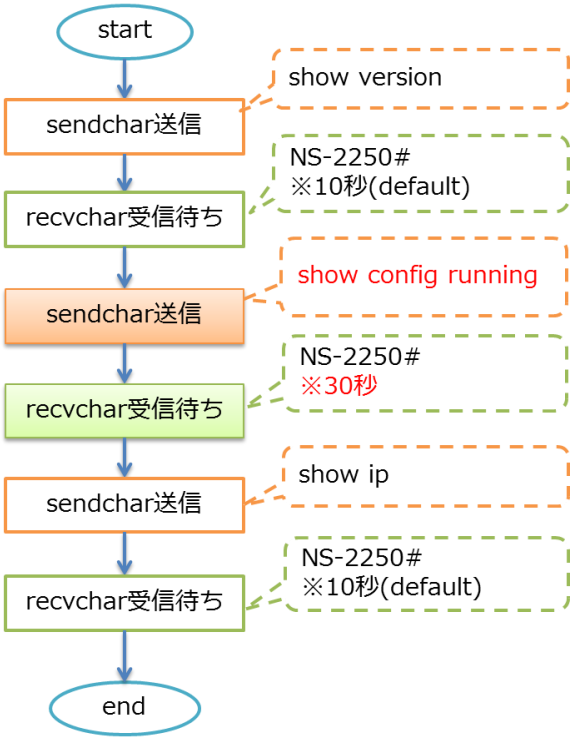
改行だけを送信したい場合に使用

SEIKO

■ 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
       recvchar :
          - 'login: '
          - 'Password: '
          - 'NS-2250>'
          - 'NS-2250#'
       sendchar :
          - show version
          - show config running __WAIT__:30
          - show ip
```



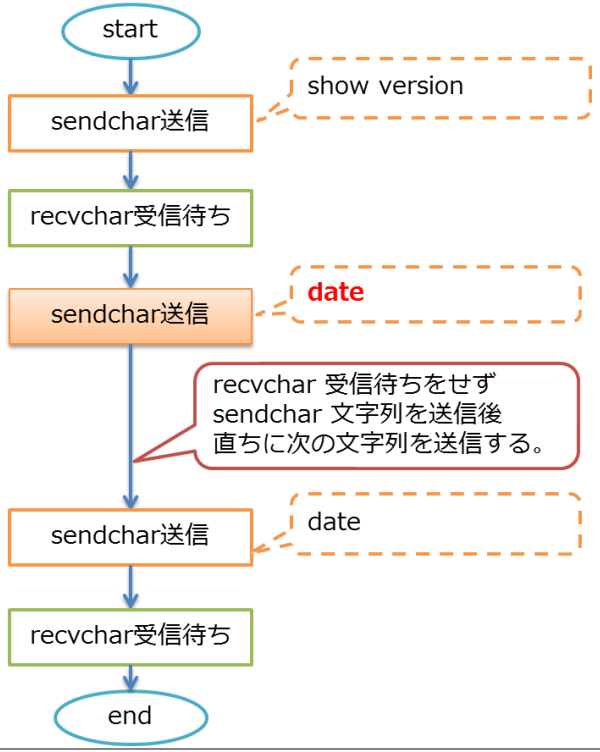
特定のsendcharを送信後、
タイムアウト時間を変更したい時に使用する事ができます。
※特定のコマンドのみ実行時間が長い場合など

SEIKO

■ 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
       recvchar :
          - 'login: '
          - 'Password: '
          - '>'
          - '#'
          - '[y/n] ? '
       sendchar :
          - show version
          - date __NOWAIT__
          - date
```



sendchar送信後、
recvcharを待たずに連続して文字列を送信したい
場合などに使う事ができます。

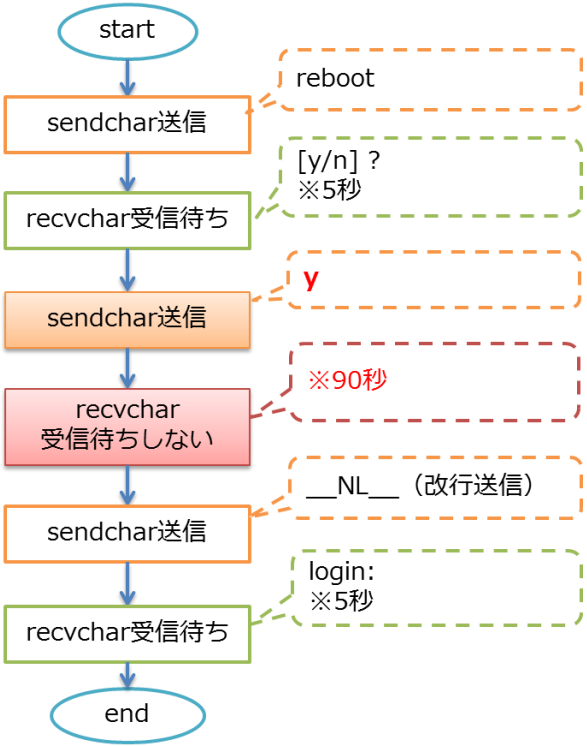
SmartCS と Ansible の連携

SEIKO

■ 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
        recvchar :
          - '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recvchar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
recvcharを受信してしまう場合など。

SEIKO

■ 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。

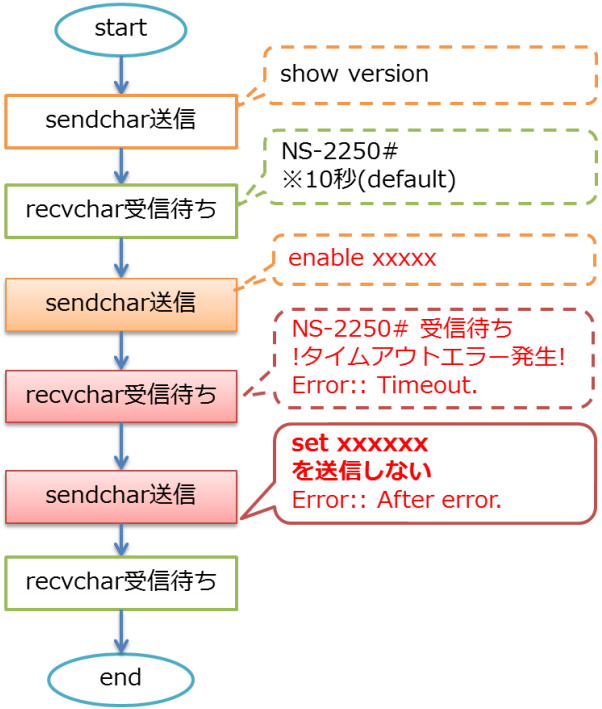
※error_recvchar_regex について
設定していない場合は、有効となりません。
設定している場合、送受信文字列に指定した文字列が含まれている場合、エラーと判定します。

SEIKO

■ 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
       recvchar :
          - 'login: '
          - 'Password: '
          - 'NS-2250>'
          - 'NS-2250#'
       sendchar :
          - show version
          - enable xxxxx
          - set xxxxxx
```



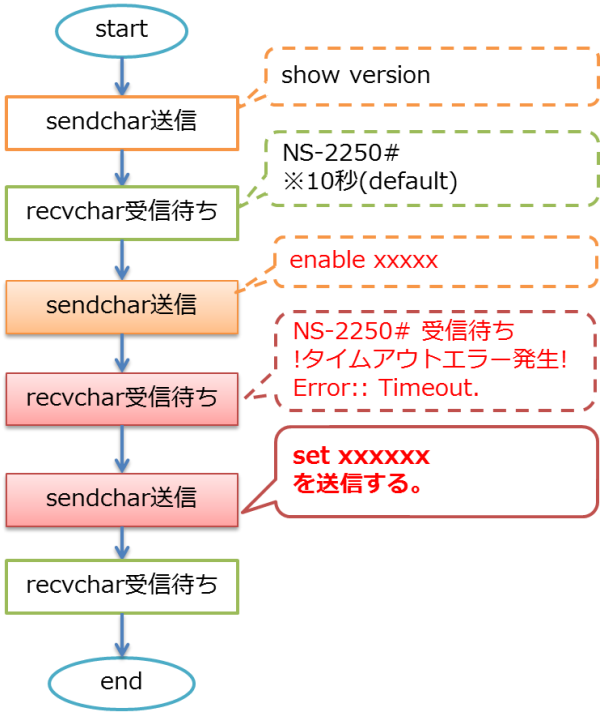
エラーが発生した場合、エラー発生後のsendcharを
送信しない場合に設定します。
※誤動作防止など

SEIKO

■ 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
```



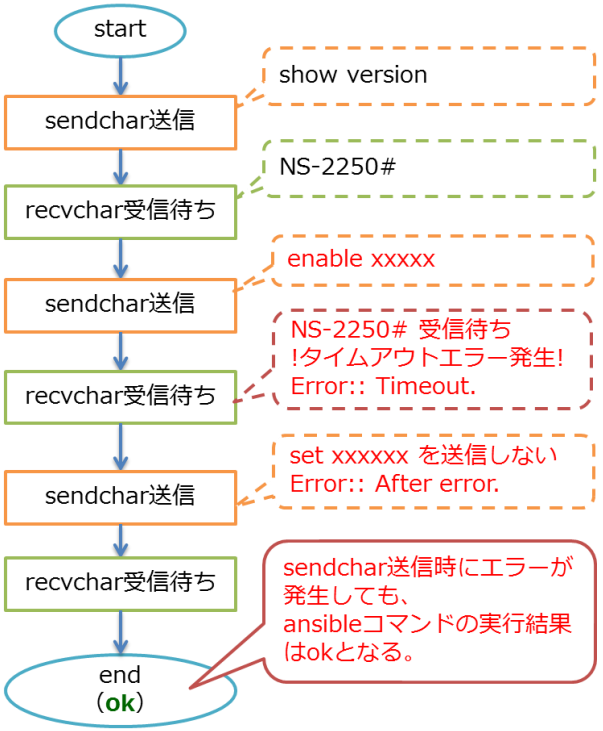
エラーが発生した場合でも、
sendcharを送信する場合に設定します。

SEIKO

■ 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
       recvchar :
         - 'login: '
         - 'Password: '
         - 'NS-2250>'
         - 'NS-2250#'
       sendchar :
         - show version
         - enable xxxxx
         - set xxxxxx
```



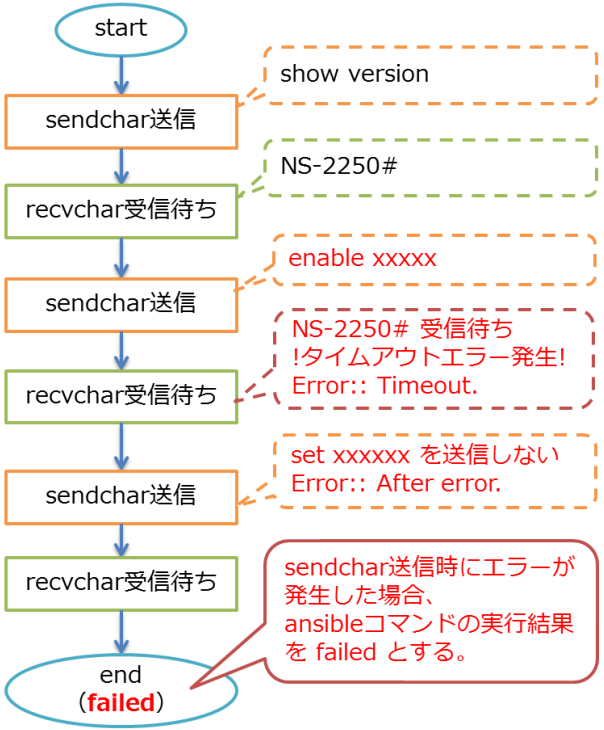
エラーが発生した場合でも、
ansibleコマンドはfailedとせず、okとする場合に設定します。
※判定可能なエラーは、
「sendchar送信後に発生するエラー要因」
の内容に限ります。

SEIKO

■ 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
        recvchar :
          - '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

SEIKO

■sendchar の送信前にコンソールの状態をチェックする機能

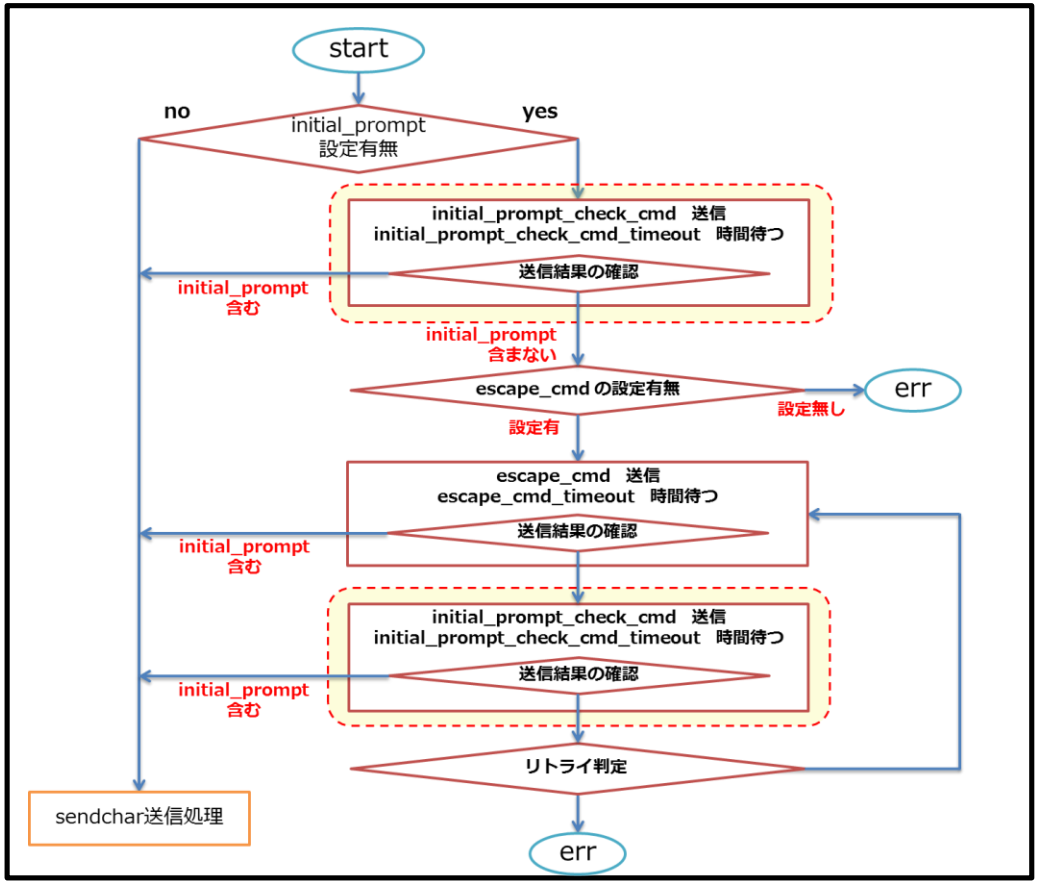
・sendchar 送信前にコンソールの状態が期待する状態かを確認します。

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 と Ansible の連携



SEIKO

■ 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

SEIKO

■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

コマンド(sendchar)実行結果

sendchar送信

recvchar受信

コマンド(sendchar)実行結果

SmartCS と Ansible の連携

SEIKO

■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

コマンド(sendchar)実行結果

recvchar受信

sendchar送信

コマンド(sendchar)実行結果

recvchar受信

SEIKO

■ 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#"
           ],
       }
]
```

SEIKO

■ 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",
           ],
        }
]
```

SEIKO

■ 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

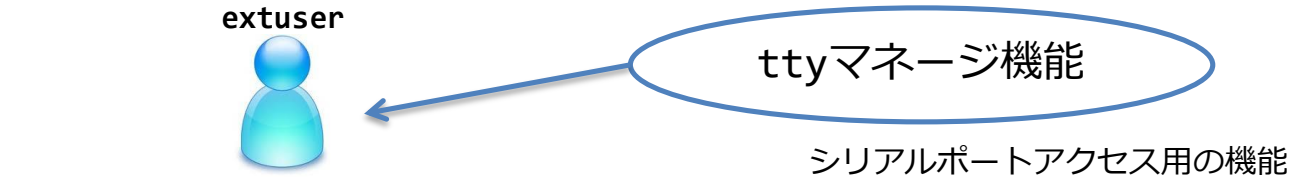
ANSIBLE

SmartCS と Ansible の連携

SEIKO

■ 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。



SEIKO

■ 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。

SEIKO

■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	○	×	○	×	○	×
拡張ユーザ	**extusr**	○	×	○※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

ANSIBLE

ベンダモジュールとの連携

■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。

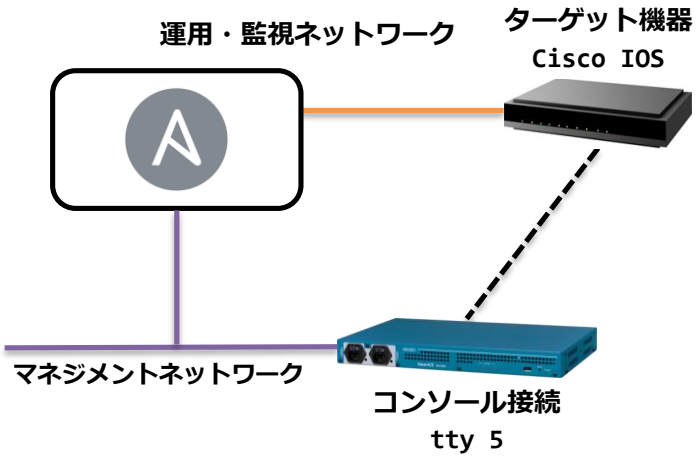
・Playbookのタスク部分については変更する事なく、再利用が可能です。

構成

SEIKO

■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

ベンダモジュールとの連携

SEIKO

■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）

SmartCS経由
装置コンソールへのログイン処理

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）

SmartCS経由
装置の制御（設定・表示）

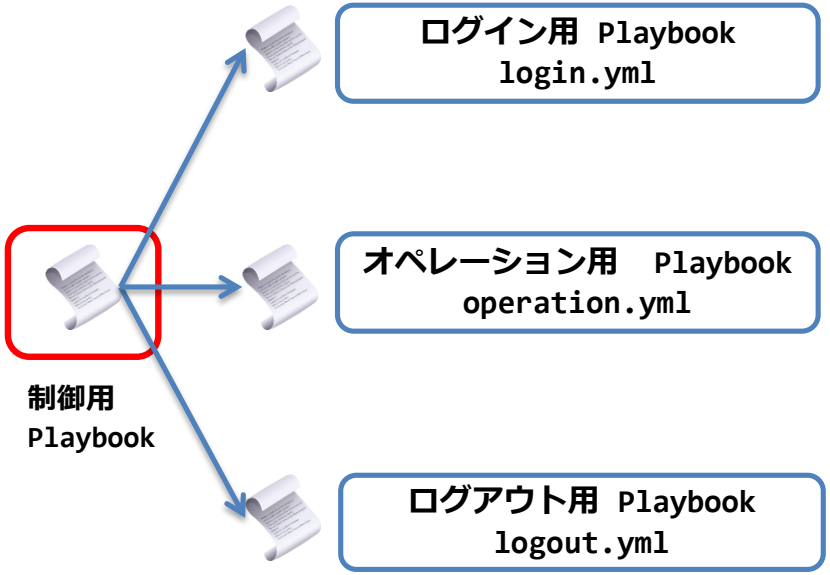
Playbook ③
Module: smartcs_tty_command

ユーザ：拡張ユーザ
ポート：SSHポート（22）

SmartCS経由
装置コンソールからのログアウト処理

ベンダモジュールとの連携

■他ベンダーモジュールの実行（Playbook構成例）



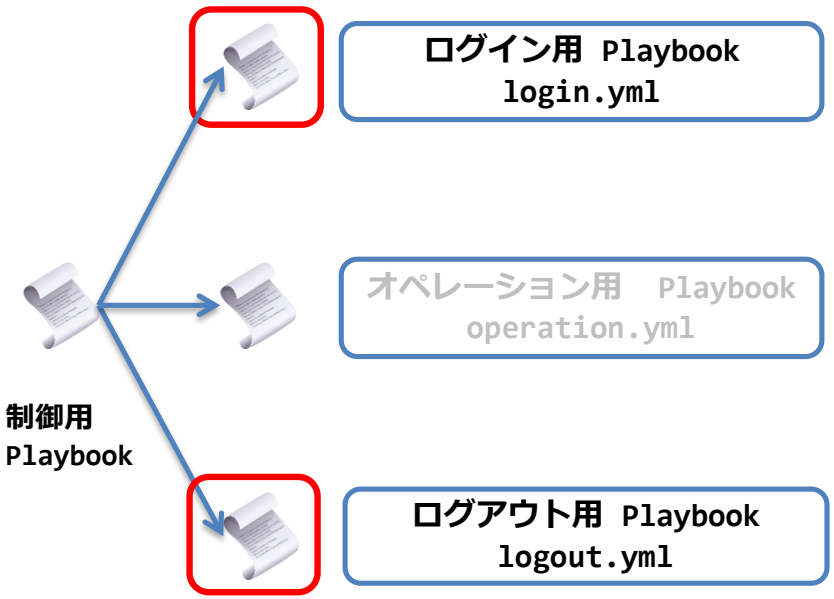

```
---
- name: "LOGIN with smartcs_tty_command"
  import_playbook: login.yml

- 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

ベンダモジュールとの連携

SEIKO

■他ベンダーモジュールの実行
(Playbook構成例)



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     ←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
```

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

logout.yml

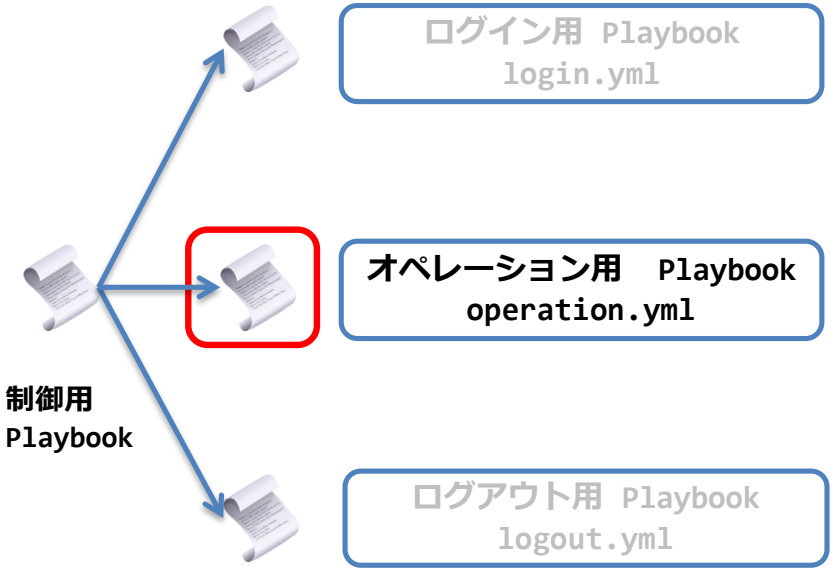
```
---
- hosts: smartcs
  tasks:
  - name: "Logout by Console"
    smartcs_tty_command:
      tty: 5
      recvchar :
      - "Press RETURN to get started.“
      recvchar_regex:
      - '(^|¥r|¥n|!)[a-zA-Z0-9_.-]*(>|#)'
      sendchar :
      - exit
```

exit 送信回数は
NW機器に応じて
複数回指定

ベンダモジュールとの連携

SEIKO

■他ベンダーモジュールの実行
(Playbook構成例)



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
```

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

SEIKO

■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
```

ベンダモジュールとの連携

SEIKO

■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 のサーボ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

ベンダモジュールとの連携

SEIKO

■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_nl { 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

SEIKO

■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/tree/2020.01.31

SEIKO

セイコーソリューションズ株式会社