

## SmartCS × IOS × Ansible ハンズオン

時間：7月29日（木）13時～17時（受付12時50分より）

場所：オンライン開催@Zoom

内容	担当
NW自動化とAnsible	エーピーコミュニケーションズ 社
コンソールサーバー SmartCSの説明	セイコーソリューションズ 社
<b>■ ハンズオン</b> 演習1：ハンズオン環境の確認 演習2：SmartCSの基本動作(手動編)	セイコーソリューションズ 社
Ansible×SmartCSについて	セイコーソリューションズ 社
<b>■ ハンズオン</b> 演習3：Ansible×SmartCS×IOSの連携 (基礎編) 演習4：Ansible×SmartCS×IOSの連携 (応用編)	セイコーソリューションズ 社
本日のまとめ	エーピーコミュニケーションズ 社

# 【NW自動化とAnsible】

エーピーコミュニケーションズ社



**Red Hat**

Ansible Automation  
Platform

## コンソールサーバー SmartCSとは

- ・コンソールサーバ SmartCS の説明
- ・ SmartCSのアクセス方法
- ・ SmartCSのその他機能について

コンソールポートとは

- NW機器のコンソールポートは  
通常以下のような用途で使われます

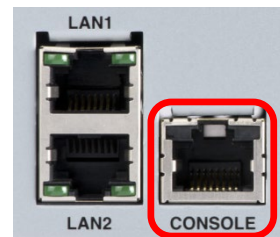
① IP設定等の**初期構築**作業

② 緊急時のオペレーション

LANインターフェース障害など、直接**IPリーチ出来ない場合**の「最後のアクセス手段」

とはいえ

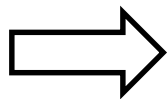
- 監視対象装置全てのコンソールポートに対してそれぞれ監視端末を用意できない
- 緊急時に現地まですぐに行くことができない



RJ45



DB9



**コンソールサーバーの出番です！**

## コンソールサーバー *SmartCS*

- コンソールポートへのアクセスをリモートから行えるようにする装置
- DC内でToR等に設置され、監視対象装置に接続してNOCからの操作を可能にします。

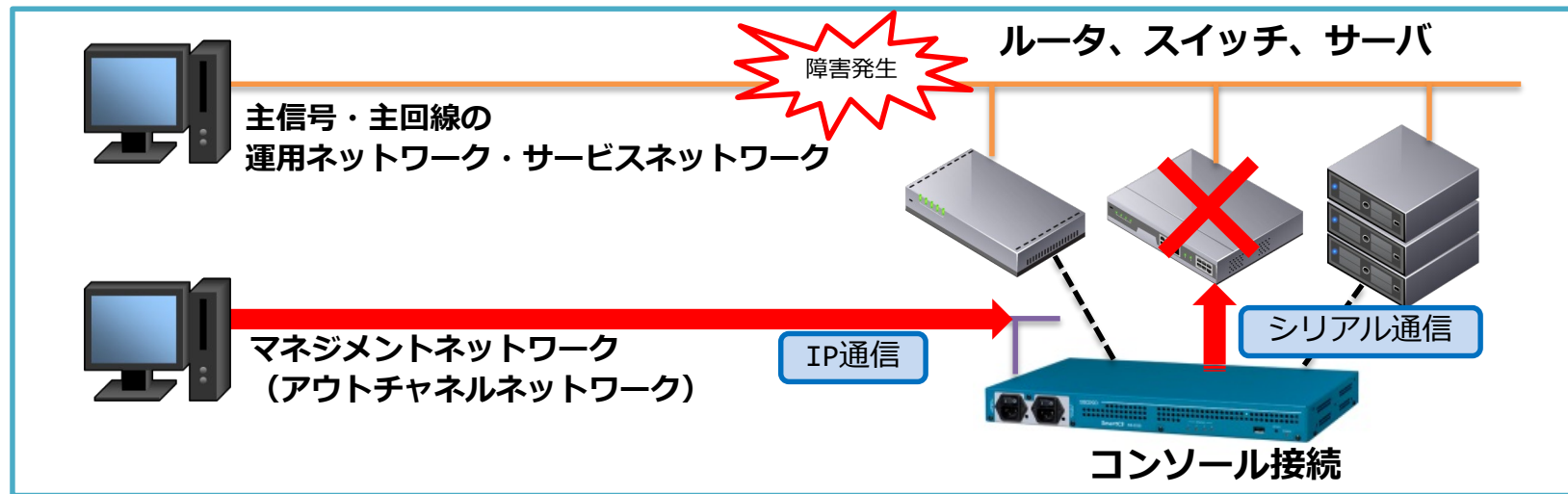


装置裏側

一見、多ポートスイッチのように見えますが、  
1台で最大48のコンソールポートを集約可能

- ・ 通信キャリア様、ISP様など大規模NWを運用しているお客様を中心に、コンソールサーバー SmartCSシリーズは国内で高いシェアを確立
- ・ INTEROP shornetのネットワーク構築においても10年以上の実績

## コンソールサーバーを利用する場合のNW構成



主回線の運用ネットワークやターゲット機器へのアクセスがNGとなった場合、コンソールサーバ経由で、監視対象装置にアクセスしてオペレーションを実行



最後のライフラインとしてアクセス手段を提供

SmartCSには、下記2通りのアクセス方法があります

## <ダイレクトモード>

SmartCSのシリアルポートに割り当てられたTCPポート番号を指定してアクセス

例：

tty1にアクセス → SmartCSの Port8301 にアクセス

## <セレクトモード>

一旦SmartCS自身へアクセスし、ポートセレクトメニューからアクセス先を選択

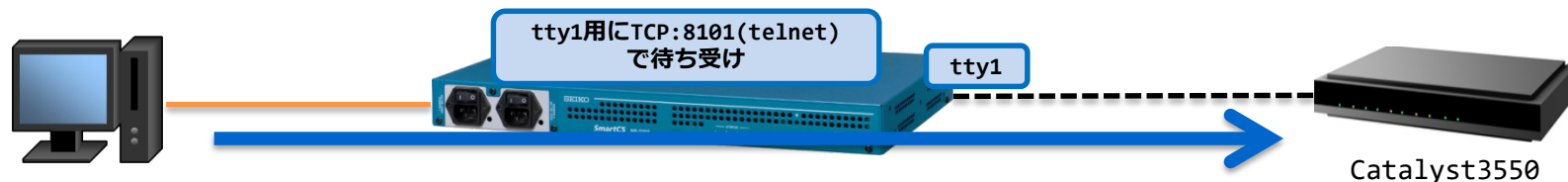
例

SmartCSにSSH(22)でアクセス後、接続したいtty番号を選択



## <ダイレクトモード>

- ・各ttyに割り当てられているTCPポートを指定するだけで、ダイレクトに接続可能
- ・接続するttyにどの機器が繋がっているかを別に管理し、把握しておく必要あり



### 【ダイレクトモードの接続イメージ】

```
$ telnet 192.168.0.1 8101
Host : "SmartCS-2250"
login from 10.208.36.40

Login: cisco
Password:
Cat3550>
.
.
.
```

telnetでtty1にダイレクトモードで接続する場合

### 【ダイレクトモードで使用するTCPポート】

tty	TCPポート(telnet)	TCPポート(SSH)
1	8101	8301
2	8102	8302
47	8147	8347
48	8148	8348

## <セレクトモード>

- SmartCSの代表ポート(telnet:23/SSH:22)に接続し、リストから選択して接続
- ラベル設定することで、各ttyにどの機器が繋がっているかをリストから把握可能
- 開いているターミナルのまま、別のttyへ操作を切り替えることが可能(切替文字 Ctrl+XX入力)



### 【セレクトモードの接続イメージ】

```
$ telnet 192.168.0.1
Login: port01
Password:

Host : "SmartCS-2250"
login from 192.168.0.254
user (port01) Access TTY List
=====
tty : Label                                RW    RO
-----
1 : Cat3550_1                             0      0
2 : Cat3550_2                             0      0
.
.
.
```

telnetでセレクトモードで  
接続する場合

### 【操作対象ttyの切り替え】

#### tty1の操作

```
Login: cisco
Password:
Cat3550_1>
Cat3550_1> show version
.
.
.
Cat3550_1> exit
Login:
```

#### SmartCSメニュー

```
Host : "SmartCS-2250"
login from 192.168.0.254
user (port01) Access TTY List
=====
tty : Label                                RW    RO
-----
1 : Cat3550_1                             0      0
2 : Cat3550_2                             0      0
```

#### tty2の操作

```
Login: cisco
Password:
Cat3550_2>
Cat3550_2> show version
.
.
.
Cat3550_2> exit
Login:
```

コンソールにアクセスする、という用途以外にも、運用管理を支援する便利な機能があります。

## <ログ保存/転送機能>



- 装置内部にログを保存するだけでなく、外部サーバへも出力が可能です

## <シリアルポートへのアクセス制限>



- ユーザ毎にアクセス可能なシリアルポートを設定できます。

## <ポートミラーリング>



- 監視対象機器への操作内容を複数のユーザで確認できます。

コンソールで入出力されるログは以下のような種類があります。

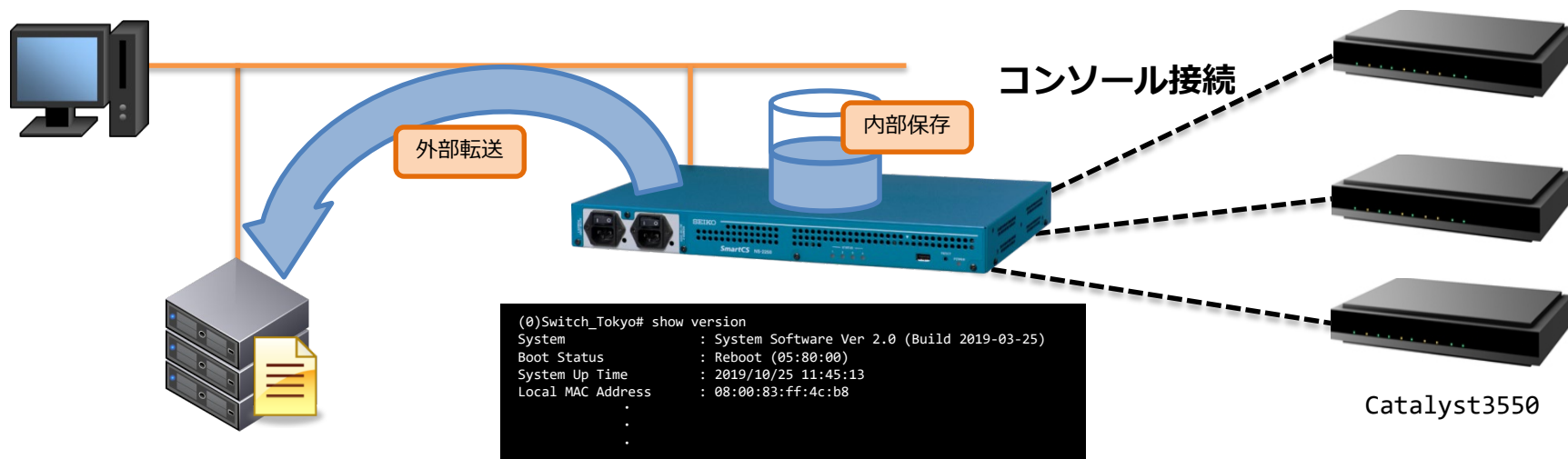
- ①装置が自発的に出力するログ
  - コンソールにしか出力されないログ
  - シャットダウン / 再起動発生時 のログ
  - 障害発生直前のエラーログ
- ②オペレーションログ
  - コンソールサーバを経由して操作したオペレーションログ

## ■ ログ保存機能

- ・ コンソールに入出力されるログを装置内部に保存します。
  - SmartCS本体に、シリアルポートごとに3Mまで保存可能(最大8Mまで拡張可能)
  - 設定不要で自動的にオペレーションログを装置内部に保存します。  
→ ログの保存忘れや誤って削除する事を防ぐことができます。

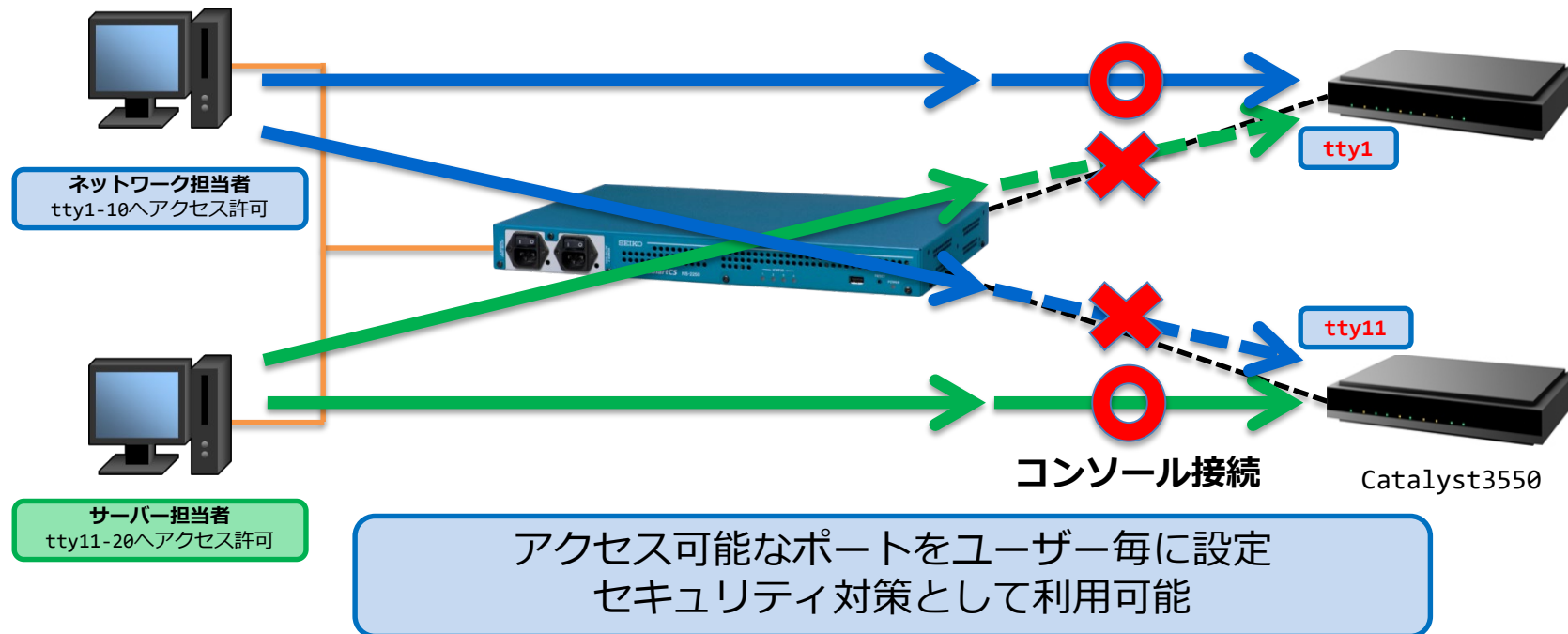
## ■ ログ転送機能

- ・ 装置内部に保存する以外にも、外部サーバへの転送が可能です
- ・ FTP / Mail
  - 送信時間 / ログの保存領域の閾値に応じた送信タイミングを指定可能
- ・ Syslog / NFS
  - ログが出力されたタイミングでログを送信



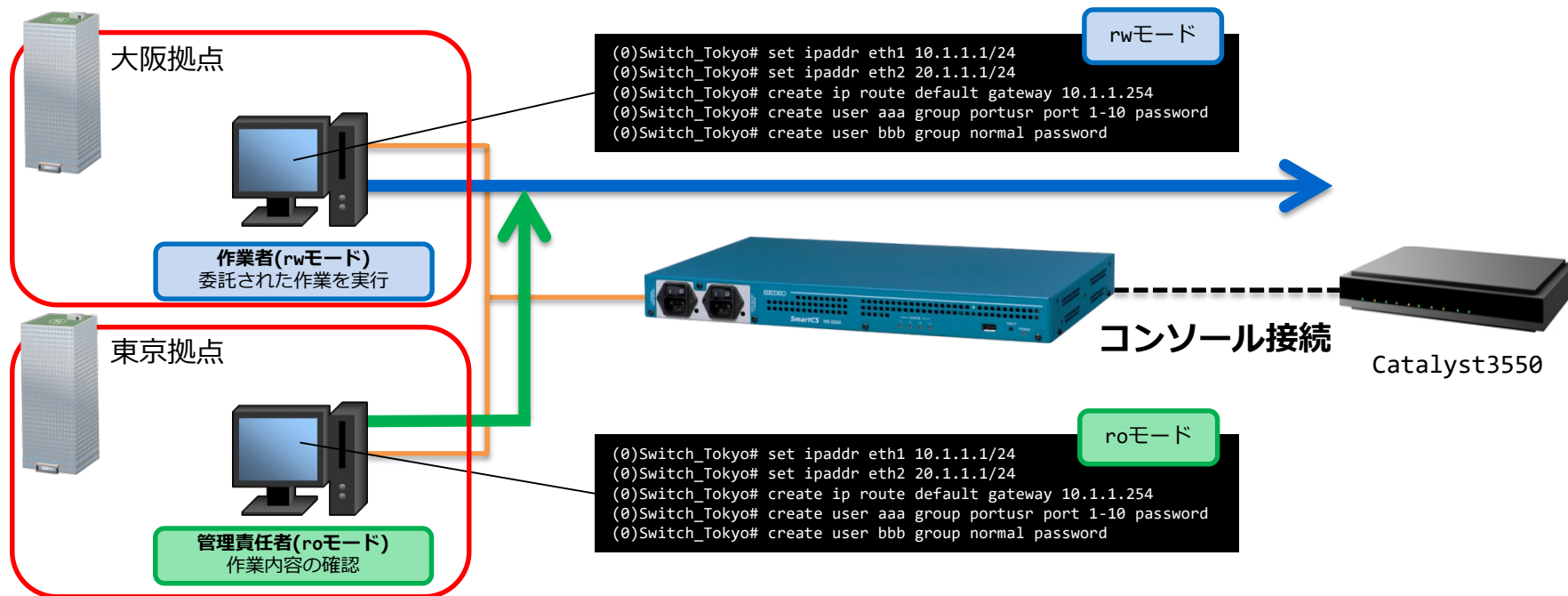
## ユーザー毎のアクセス制限

- ミスオペレーションによる操作対象間違い
- 権限の無い機器への不正アクセス などを防止します。



特定のシリアルポートへの操作内容を複数のユーザで確認することができます。

- rw権限：送受信可能なモードで、監視しつつ制御も可能
- ro権限：受信のみ可能なモードで、監視のみ可能





## 演習内容

### 演習1 ハンズオン環境の確認

- 1.1 演習環境の確認

### 演習2 SmartCSの基本動作(手動編)

- 2.1 SmartCSを介してIOS装置へコンソールアクセスする
- 2.2 SmartCSを介したIOS装置へのコンソールアクセスをミラーリングする
- 2.3 SmartCSを介したシリアルセッション情報を確認する

### 演習3 Ansible×SmartCS×IOSの連携演習(基礎編)

- 3.1 IOS装置にSmartCS経由で初期設定を行う
- 3.2 IOS装置に追加設定を行う
- 3.3 IOS装置の設定情報を取得する
- 3.4 IOS装置の設定情報をSmartCS経由で取得する

### 演習4 Ansible×SmartCS×IOSの連携演習(応用編)

- 4.1 オペレーションミスからの復旧自動化
- 4.2 通信障害からの復旧自動化
- 4.3 初期化の自動化

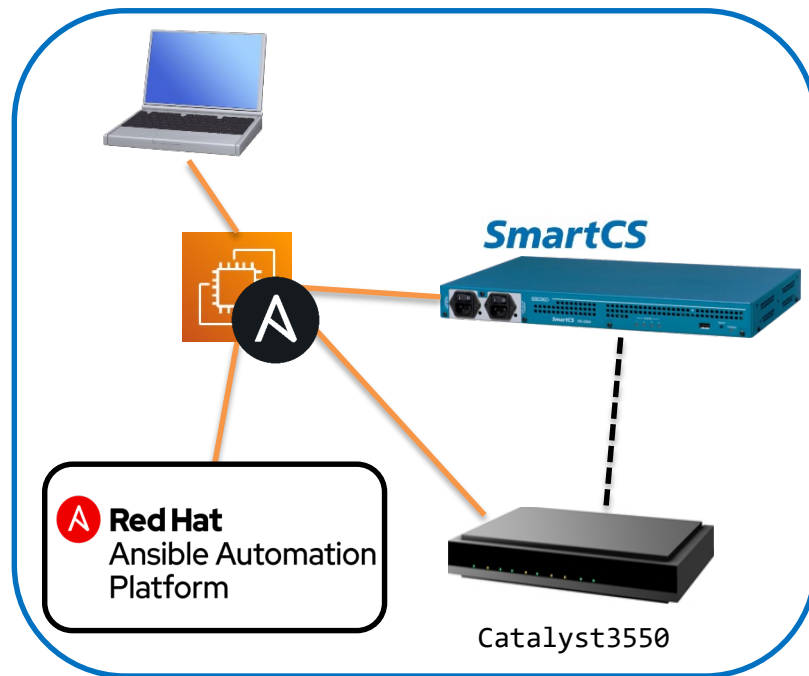
# 【ハンズオン 演習1】

## ハンズオン環境の確認

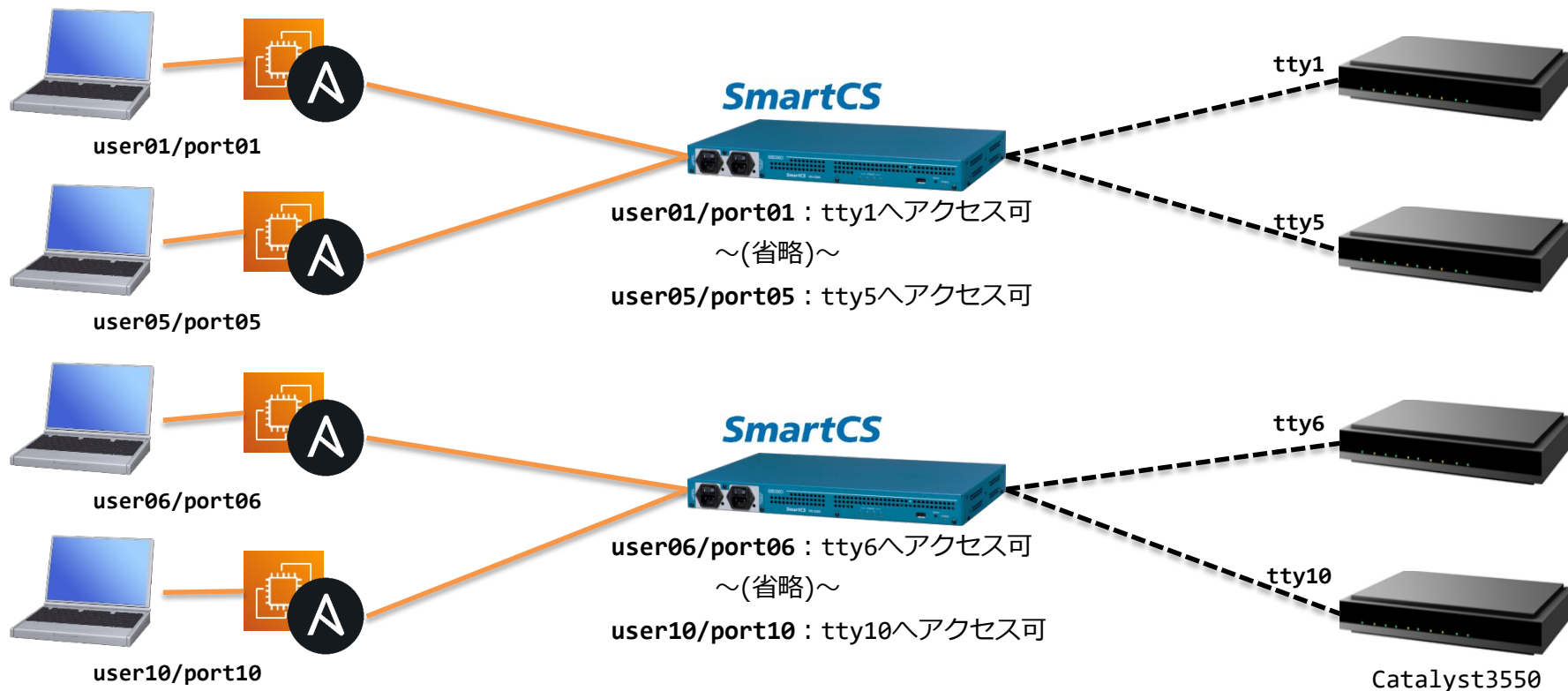
## 環境概要

- 参加者1名に対して1つの環境を用意しています。  
(SmartCSは5名で1台を使用いただきます。)
- SmartCS(1台目)のtty1, 2, 3, 4, 5と、  
SmartCS(2台目)のtty6, 7, 8, 9, 10に、  
Catalyst3550のコンソールが接続されています。
- 演習2では、EC2(Ansibleノード)にSSHでアクセスしてから  
SmartCSにTelnet/SSHでアクセスいただき、  
SmartCS経由でのコンソール操作を体感していただきます。
- 演習3以降では、EC2(Ansibleノード)にSSHでアクセスいただき、  
Catalyst3550の操作を実施していただきます。

## 構成



参加者ごとに割り当てられているアドレス/ユーザ/パスワードを利用します。(手順書の演習1を参照)  
SmartCSを介してアクセスする場合、各ユーザ(userXX/portXX)には、アクセス可能なttyが設定されています。



**【ハンズオン 演習2】**

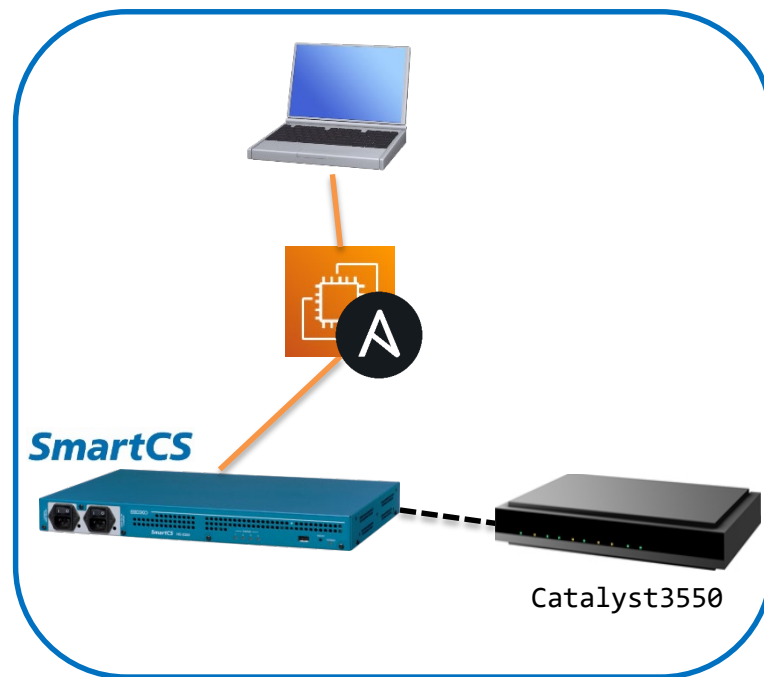
**コンソールサーバー SmartCSの基本動作(手動編)**

## 概要

演習2では、SmartCSの基本的な使い方を確認します。

- SmartCSを経由して、Catalyst3550のコンソールへアクセス【演習2.1】
- SmartCSを経由したセッションをミラーリング【演習2.2】
- SmartCS上でシリアルセッション情報の確認【演習2.3】

## 構成



## ■ハンズオン手順書(Github)

<https://github.com/ssol-smartcs/ansible-handson/SmartCS×IOS>

**【座学】**

**Ansible × SmartCS について**



## 背景 ネットワーク運用環境の変化

手動オペレーション

運用の自動化

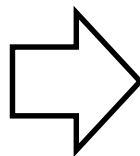
CLI / GUI

外部API

Orchestrator

運用ツール

**SmartCS**



運用自動化への対応

## 従来の運用自動化における課題

- リモートからの設定変更により、機器へリモートから接続できなくなる可能性
- リモートからのバージョンアップ作業(失敗)により、通信できなくなる可能性
- データセンタへ駆けつけ、機器のコンソールへPCを直結し復旧しなければならない

## SmartCSによる解決

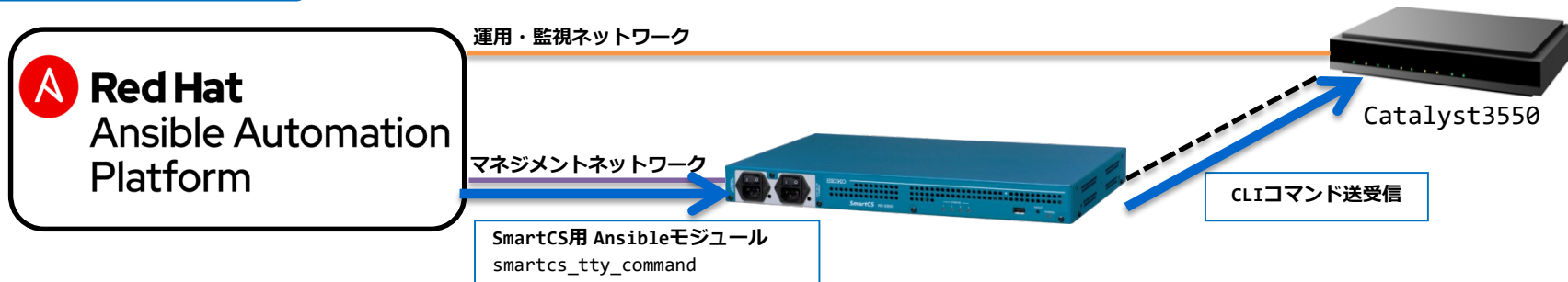


SmartCSがあれば、リモートからコンソールをオペレーション可能！

## 従来のAnsibleにおける課題

- Ansibleリーチできない状態の機器の操作が難しい (初期設定段階)
- Ansibleモジュールが無い機器の操作にはあまり適していない (ベンダー依存)

## SmartCSによる解決



ConsoleからCLI操作可能な機器は、Ansibleによるオペレーション自動化の対象に！

## Ansible をさらにパワフルに

従来は ネットワーク機器・サーバ機器 等のターゲットが  
IPリーチ（Ansibleリーチ）可能になっている状態でないと  
各モジュールによるオペレーションが実行出来なかった

Ansible をさらにパワフルに



IPリーチャビリティのないターゲットも運用自動化の対象に  
+  
Ansibleモジュールのないターゲットも対象に

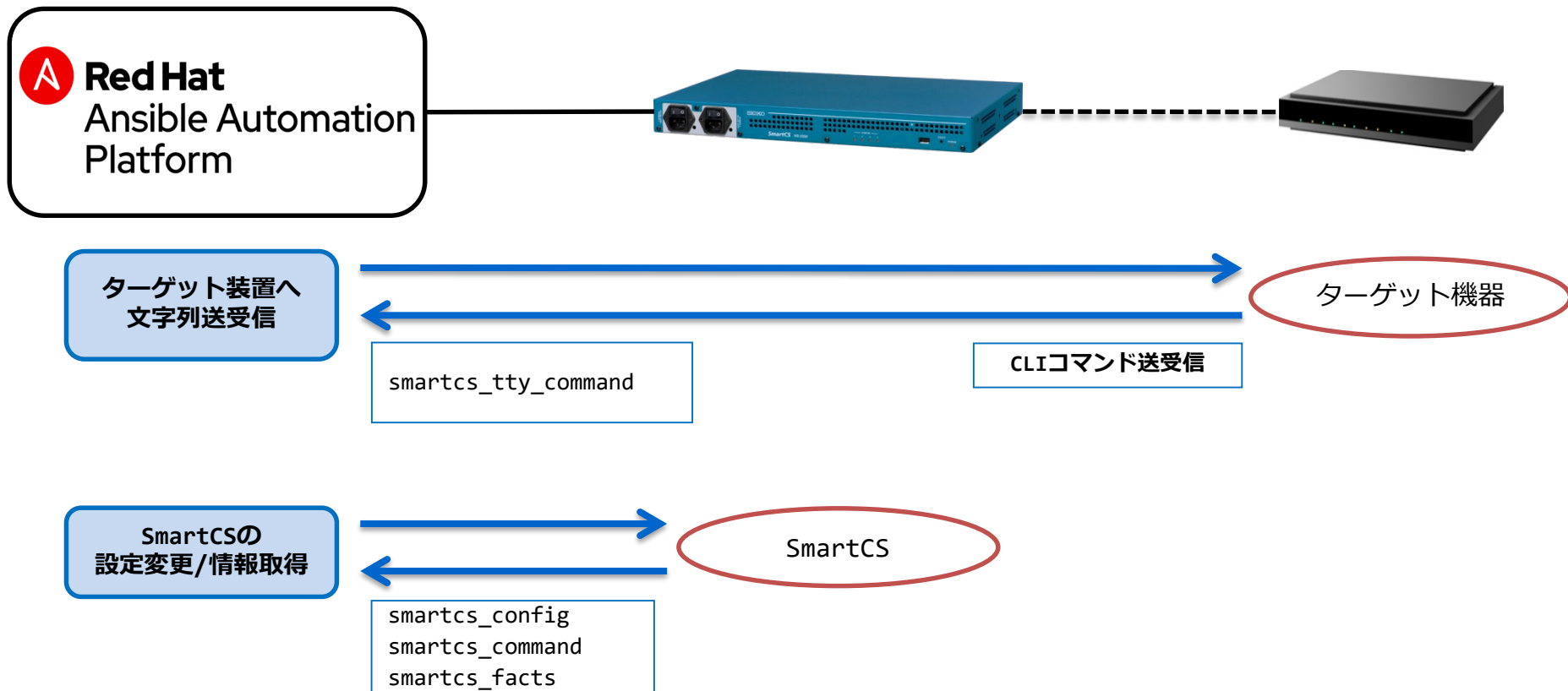
SmartCS用のAnsibleモジュールをAnsible Engineにインストールすることで以下のオペレーションをAnsible経由で行うことが可能となります。

- SmartCSのシリアルポートに接続されている監視対象機器にして、Ansibleから文字列(対象機器のコマンド)を送受信することが可能  
→ `smartcs_tty_command`

本日のハンズオン  
内容

- SmartCS自身の設定変更、および情報取得が可能  
→ `smartcs_facts`, `smartcs_command`, `smartcs_config`

リモートからコンソール経由での設定変更、バージョンアップなどの作業をAnsibleで自動化することができるようになります。



v1.3.0以降のモジュールはAnsible Collectionsに対応しており、Ansible Galaxyから取得してインストール可能です。

- Ansible Galaxy

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

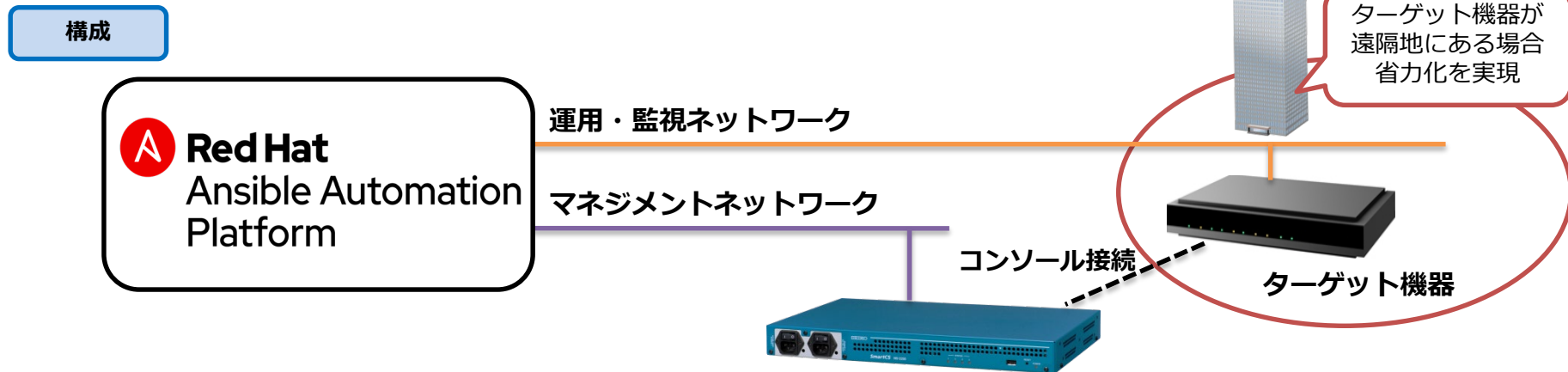
The screenshot shows the Ansible Galaxy interface for the 'seiko.smartcs' collection. The page is titled 'SmartCS smartcs' and describes it as an 'Ansible Network Collection for Seiko SmartCS devices'. It shows 65 downloads and provides links to login, issue tracker, repository, website, and documentation. The 'Details' tab is selected, showing the installation command: `$ ansible-galaxy collection install seiko.smartcs`. A note states that installing collections with ansible-galaxy is only supported in ansible 2.9+. The 'Install Version' is 1.4.0, released 7 days ago. The 'Tags' are seiko, smartcs, and networking. The 'Content Score' section shows a community score of 0/5 based on 0 surveys. The 'Tell us about this collection' section includes a quality of docs rating and a table for other metrics.

Tell us about this collection	
Quality of docs?	- ○ ○ ○ ○ +
Ease of use?	- ○ ○ ○ ○ +
Does what it promises?	Y N
Works without change?	Y N
Ready for production?	Y N



## 初期構築（設置時・交換時）

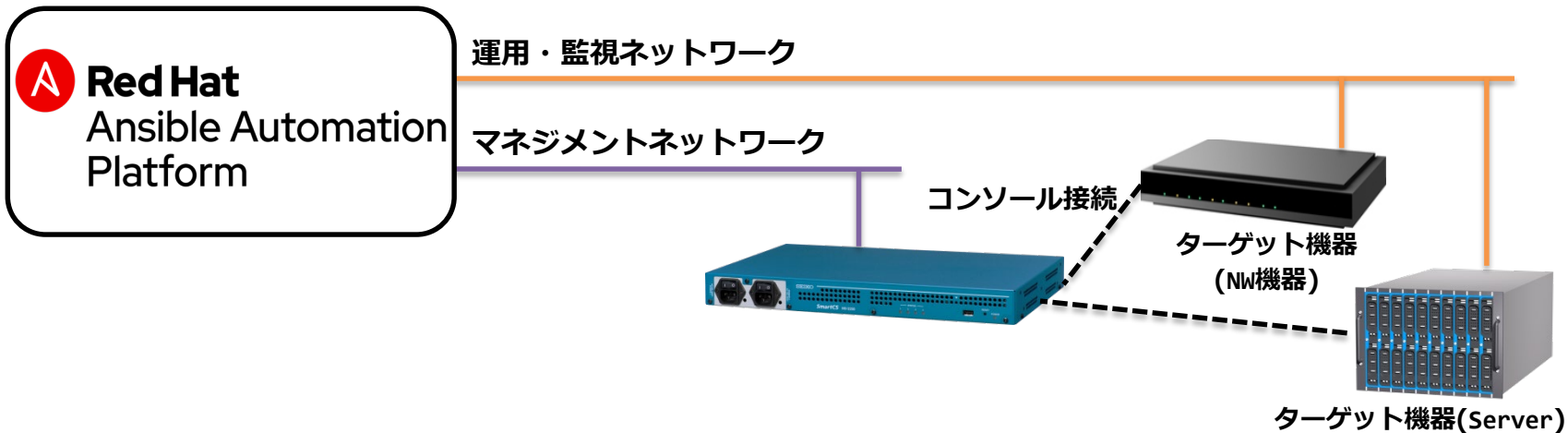
NW機器の設置時や交換時に、コンソール経由で**IP設定/ユーザ作成**などの初期構築を行います。  
最低限の初期設定をコンソール経由で実行し、Ansibleリーチ可能な状態になってからは  
ベンダーごとに用意しているモジュールを利用して追加の設定等を行います。



## コンソールからのバージョンアップ作業

NW機器やサーバ（Hypervisorのホスト）機器の設定変更やバージョンアップ作業を、コンソール経由で安全に行います。

### 構成



SmartCS用のAnsibleモジュール“smartcs\_tty\_command”では、  
下記の様なパラメータをplaybook内で指定して文字列の送受信を行います。

演習で使用

パラメータ名	設定値	概要
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>c</u> r / l <u>f</u> / c <u>r</u> l <u>f</u>	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。

SmartCS用のAnsibleモジュール“smartcs\_tty\_command”では、  
下記の様なパラメータをplaybook内で指定して、送信文字列と受信文字列を区別しやすい  
返り値(stdout\_lines\_custom)とすることができます。

## 演習で使用

パラメータ名	設定値	概要
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。

**参考情報(v1.0)**

パラメータ名	設定値	概要
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。

**参考情報(v1.1)**

パラメータ名	設定値	概要
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。

```
---
- name: Login Catalyst3550
  hosts: smartcs
  gather_facts: no

  tasks:
    - name: Login Catalyst3550
      seiko.smartcs.smartcs_tty_command:
        tty: 1
        nl: cr
        cmd timeout: 5
        recvchar:
          - "# "
          - "> "
          - "(config)# "
          : 省略
        sendchar:
          - __NL__
          - enable
          - configure terminal
          : 省略

  vars:
    - ansible_command_timeout: 60
    - ansible_connection: ansible.netcommon.network_cli
    - ansible_network_os: seiko.smartcs.smartcs
    - ansible_user: user01
    - ansible_password: secret01
```

## ■recvchar (recvchar\_regex)

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

## ■sendchar

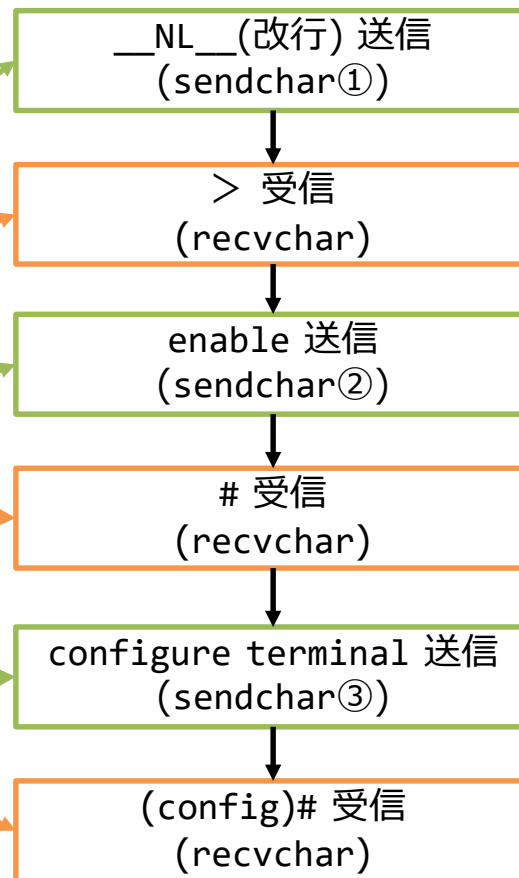
- ・ 指定した tty に送信する文字列を指定します。
- ・ リストの上から順番に送信します。

## ■vars

- ・ **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かかります。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**  
→ **ansible.netcommon.network\_cli** を指定します。
- ・ **ansible\_network\_os**  
→ **seiko.smartcs.smartcs** を指定します。
- ・ **ansible\_user** , **ansible\_password**  
→ SmartCSにログインする為の**拡張ユーザ(extusr)**のログイン情報を指定します。

## sendcharとrecvcharの動作

```
tasks:
- name: Login Catalyst3550
  seiko.smartcs.smartcs_tty_command:
    tty: 1
    nl: cr
    cmd_timeout: 5
    recvchar:
      - "# "
      - "> "
      - "(config)# "
    sendchar:
      - __NL__
      - enable
      - configure terminal
```





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\nCisco IOS Software, C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(44)SE6, RELEASE SOFTWARE (fc1)\n\nCopyright (c) 1986-2009 by Cisco Systems, Inc.\n\n(省略)\n\nCat3550>"
],
```

送信したsendchar

受信したrecvchar

## stdout\_lines出力例

```
"stdout_lines": [
  [
    "show version",
    "",
    "",
    "",
    "Cisco IOS Software, C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(44)SE6, RELEASE SOFTWARE (fc1)",
    "",
    "Copyright (c) 1986-2009 by Cisco Systems, Inc.",
    "",
    "(省略)",
    "",
    "",
    "",
    "Cat3550>"
  ]
],
```

送信したsendchar

受信したrecvchar

コマンド  
(sendchar)  
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": [
      "Cisco IOS Software, C3550 Software (C3550-
      IPSERVICESK9-M), Version 12.2(44)SE6, RELEASE
      SOFTWARE (fc1)",
      "Copyright (c) 1986-2009 by Cisco Systems, Inc.",
      (省略)
      "Cat3550>"
    ]
  }
]

```

送信したsendchar

受信したrecvchar

コマンド  
(sendchar)  
実行結果

## ■ SmartCS経由で、Ansibleの他ベンダーモジュールを利用可能

※演習3.4以降で本機能を利用した演習を実施いたします。

smartcs\_tty\_commandのみを利用

- ・ベンダー製のAnsibleモジュールがないターゲットにアクセスする場合
- ・smartcs\_tty\_commandモジュールを使って全ての制御を完結させたい場合  
→ 1つのPlaybookで全ての処理を行いたい場合

【課題】 Playbookの作成が難しい

→ 実施したい操作のコンソール経由の入出力情報(特にrecvchar)が必要、冪等性担保×

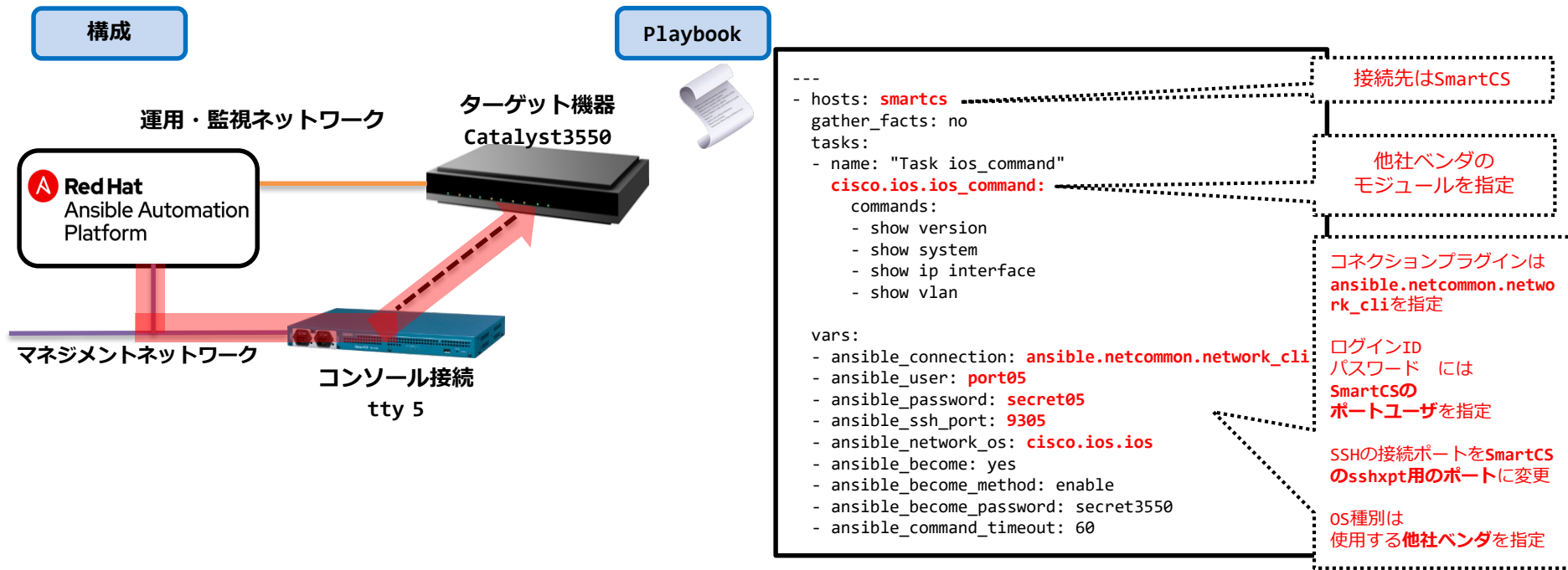
smartcs\_tty\_command と 他ベンダーモジュールを連携して利用

- ・ベンダー製モジュールを利用してターゲット機器の制御を行いたい場合

【メリット】 Playbookの作成が比較的容易

→ ベンダー製モジュールを利用したPlaybookがそのまま流用出来る、冪等性担保○

## ■ SmartCS経由で、Ansibleの他ベンダーモジュールを利用する場合の 接続構成とPlaybookイメージ



## ■ 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

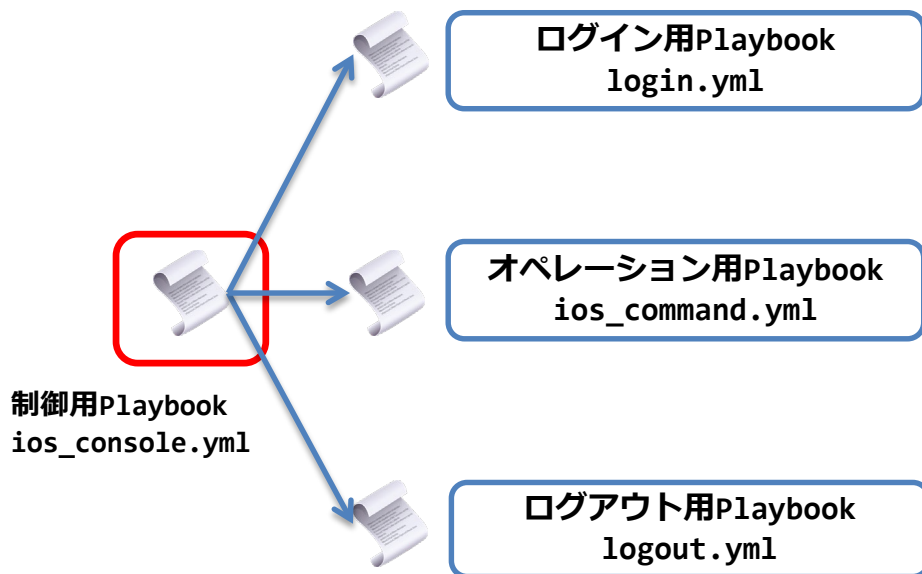


SmartCS経由  
装置の制御（設定・表示）



SmartCS経由  
装置からのログアウト処理

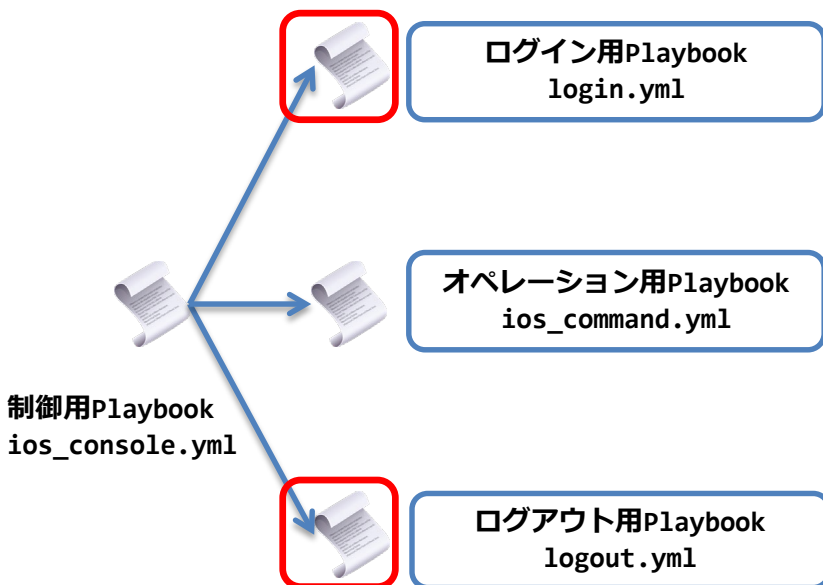
## ■ 他ベンダーモジュールの実行（Playbook構成例）



制御用Playbook例 **ios\_console.yml**  
※実際に実行するPlaybook

```
---  
- name: "LOGIN with smartcs_tty_command"  
  import_playbook: login.yml  
  
- name: "Exec Task with ios_command"  
  import_playbook: ios_command.yml  
  
- name: "LOGOUT with smartcs_tty_command"  
  import_playbook: logout.yml
```

## ■ 他ベンダーモジュールの実行 (Playbook構成例)



```
---
- hosts: smartcs
  tasks:
    - name: "Login by Console"
      seiko.smartcs.smartcs_tty_command:
        tty: 5
        recvchar_regex:
          - '[Uu]sername: '
          - '[Pp]assword: '
          - '^(^|¥r|¥n|!)[a-zA-Z0-9_.-]*(>|#)'
        sendchar :
          - cisco      ←ios装置へのログインID
          - secret3550 ←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

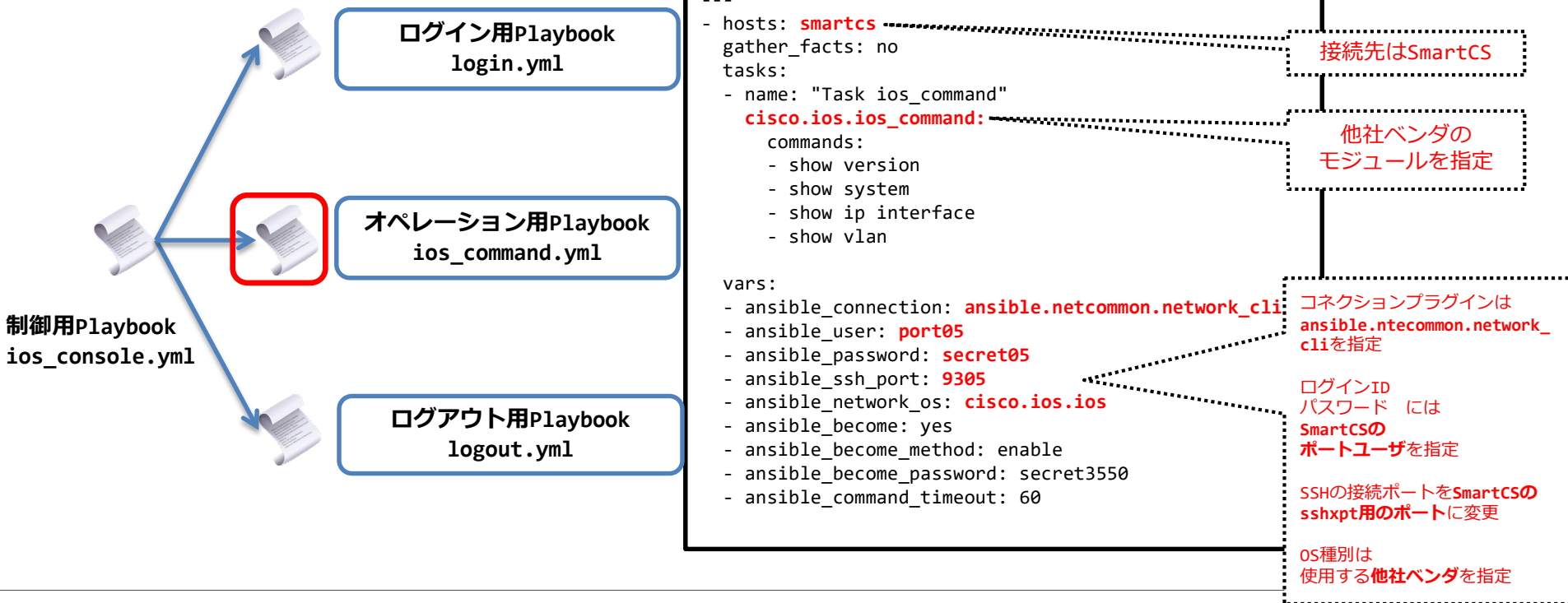
NW機器の汎用的な  
プロンプト例

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

```
---
- hosts: smartcs
  tasks:
    - name: "Logout by Console"
      seiko.smartcs.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

## ■ 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実行時のポイント

### ・ 利用できるモジュール

- ・ 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、コネクションプラグインとして**network\_cli**をサポートしているものに限りです。

例

```
vars:
  - ansible_connection: ansible.netcommon.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）

### ・ タイムアウト値の設定

- ・ 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
```

## 【ハンズオン 演習3】

### Ansible×SmartCS×IOSの連携演習(基礎編)

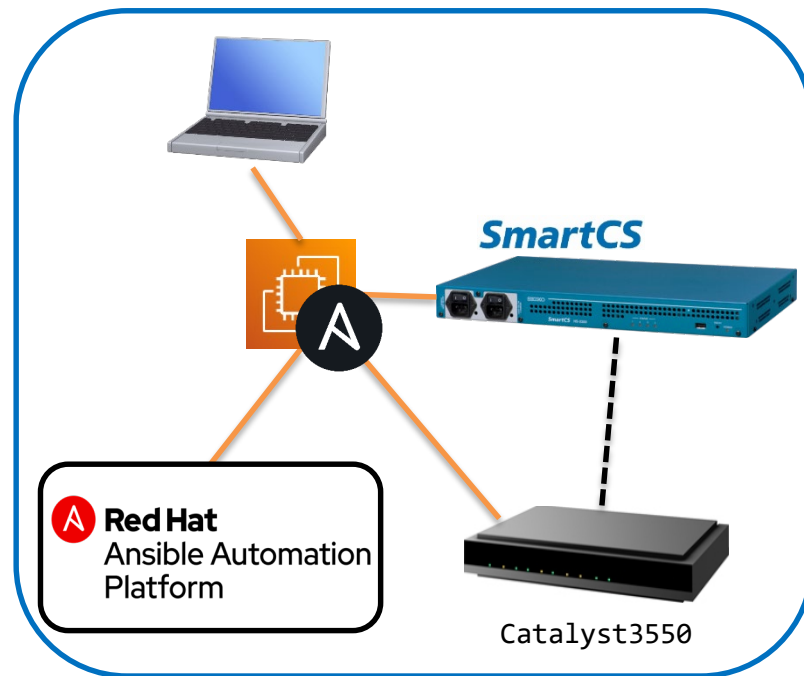
## 概要

演習3では、

- ・ SmartCSを経由して設定投入/情報取得を行い、AnsibleとSmartCSの連携方法について理解を深めていただきます。
- ・ Ansibleを使ったネットワーク機器へのオペレーションについて理解を深めていただきます。

- SmartCSモジュールでCatalyst3550への初期設定投入(SmartCS経由)【演習3.1】
- IOSモジュールでCatalyst3550への追加設定投入【演習3.2】
- IOSモジュールでCatalyst3550から設定情報取得【演習3.3】
- IOSモジュールでCatalyst3550から設定情報取得(SmartCS経由)【演習3.4】

## 構成



## 【ハンズオン 演習4】

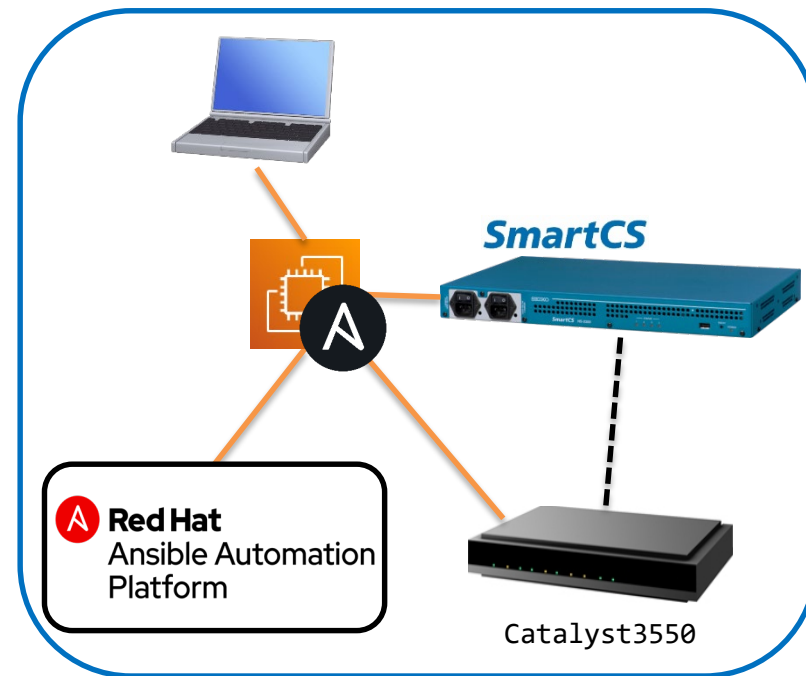
### Ansible×SmartCS×IOSの連携演習(応用編)

## 概要

演習4では、コンソールアクセスが必要となるユースケースに沿って、SmartCSとAnsibleの連携方法の理解を深めていただきます。

- 設定ミスによる障害からの復旧自動化【演習4.1】
- 通信障害からの復旧自動化【演習4.2】
- 設定初期化の自動化【演習4.3】

## 構成



**【本日のまとめ】**  
**エーピーコミュニケーションズ様**