SEIKO

Operation Guide

SmartCS modules for Ansible

Second Edition
Sep 17, 2021
U00144675003

SEIKO SOLUTIONS INC.

U00144675002 Jun 2021
U00144675003 Sep 2021

©Seiko Solutions Inc. 2019

Unauthorized reproduction is prohibited.
The contents of this document may be changed without notice.

"SEIKO" is a registered trademark of Seiko Holdings Corporation.
"Red Hat" and "Ansible" are registered trademarks or trademarks of Red Hat, Inc. and its subsidiaries in the U.S. and other countries.
"Python" is a registered trademark or trademark of the Python Software Foundation.

Seiko Solutions does not bear any liability for damage occurring due to the use of this document and the software described within or for the expenses required to recover from such damage.

Table of Contents

1 Introduction 1

1.1 “SmartCS modules for Ansible” Overview 1

1.2 Functional overview 2

1.2.1 Console Access Function Overview 3

1.2.2 CLI Command Function Overview 4

1.3 Operating environment 5

1.3.1 Control node / Managed node 5

1.3.2 Operating requirements 6

1.3.3 Ansible environment 8

1.4 License 9

1.5 How to get “SmartCS modules for Ansible” 10

2 Installation 13

2.1 Pre-installation Check 13

2.2 Installation 14

2.3 Upgrading 20

2.4 Deleting the installed collection files 21

2.5 About dependent packages 22

2.6 Other 23

3 Preparations 24

3.1 Preparing the SmartCS 24

3.2 Preparation for control node 25

4 Creating a Playbook Compatible with Ansible Collections 26

4.1 Module specification 26

4.2 Connection plugin (network_cli) specification 29

5 Console Access Function 30

5.1 Preparing the SmartCS 30

5.2 Preparation for creating a Playbook 32
5.3 Playbook Example 33
6 Linking with Network Device Vendor Modules 34
6.1 Overview 34
6.2 Preparation for SmartCS 35
6.3 Preparation for creating a Playbook 38
6.4 Playbook example 42
6.5 Instructions and Directions for Use 45
7 CLI Command Function 46
7.1 Preparing the SmartCS 46
7.2 Preparations for creating Playbook 47
7.3 Module and privileged user 48
7.4 Playbook Example 49
8 Modules 50
8.1 seiko.smartcs.smartcs_tty_command 50
8.1.1 Overview 50
8.1.2 Options 51
8.1.3 Playbook Example 60
8.1.4 Return Values 61
8.1.5 Explanations 63
8.1.6 Instructions and Directions for Use 86
8.2 seiko.smartcs.smartcs_command 88
8.2.1 Overview 88
8.2.2 Options 89
8.2.3 Playbook Example 90
8.2.4 Return Values 92
8.3 seiko.smartcs.smartcs_config 93
8.3.1 Overview 93
8.3.2 Options 94
8.3.3 Playbook Example 96
8.3.4 Return Values 98
8.4 seiko.smartcs.smartcs_facts 99

8.4.1 Overview 99
8.4.2 Options 99
8.4.3 Playbook Example 100
8.4.4 Return Values 101

9 Limitations 104

9.1 Administering the SmartCS series console with “smartcs_tty_command” 104

9.2 Gathering device information with gather_facts 105

10 Troubleshooting 107

10.1 “Unable to connect to port 22 on x.x.x.x” 107

10.2 “timed out” 107

10.3 “Error reading SSH protocol banner” 108

10.4 “The authenticity of host ‘x.x.x.x’ can’t be established.” 108

10.5 ”Authentication failed.” 108

10.6 “Bad authentication type” 109

10.7 “Unable to automatically determine host network os.” 109

10.8 “unable to elevate privilege to enable mode” 109

10.9 “command timeout triggered, timeout value is X secs.” 110

10.10 “timeout value X seconds reached while trying to send～” 110

10.11 "Ignoring timeout(10) for smartcs_facts" 112

11 Appendix A. Building the Ansible Environment 113

11.1 Building the Ansible environment with venv 113

11.2 Preparation for ansible.cfg 116

12 Appendix B. v1.0 to v1.2 Operation 117

12.1 Overview of the v1.0 to v1.2 operation 117

12.2 Pre-installation Check 117

12.3 Installation 118

12.4 Upgrading 120

12.5 Uninstalling 120

12.6 Command Reference (install_smartcs_modules) 122

13 Appendix C. Handling of Various Characters in Playbooks 124
13.1 Specifiable Character Types 124
13.2 Sending Various Types of Characters 129
13.3 Configuring Regular Expressions 132
13.4 Execution Result Output Characters 134
14 Appendix D. Tips for using the “SmartCS modules for Ansible” 135
14.1 How to Write the File Specifying the src Option 135
14.2 Sending Characters Simultaneously to Multiple Network devices 136
15 Licenses 137
15.1 Third-party Software Licenses 137
15.2 Ansible Collections package creation 156

1 Introduction

1.1 “SmartCS modules for Ansible” Overview

SmartCS modules for Ansible is the generic name for a package which uses the Red Hat Ansible Automation Platform (hereinafter, "Ansible") automation tool provided by Red Hat, Inc. that includes the modules and plugins required to operate the SmartCS console server.

Operating SmartCS via Ansible enables you to implement operations from the console including initial build work such as the IP configuration of devices connected to SmartCS via Ansible, which makes it possible to further reduce the operational load of IT infrastructure within each phase of operation.

SmartCS modules for Ansible supports Ansible Collections starting from v1.3.0.
This document primarily covers the details in accordance with the providing format of Ansible Collections.

*From v1.0 until v1.2, the modules have been provided in our original package format. For more details, refer to "12 Appendix B. v1.0 to v1.2 Operation."

This Operation Guide is a document which describes how to use and operate SmartCS modules for Ansible.

1.2 Functional overview

SmartCS modules for Ansible uses Ansible to not only configure SmartCS and obtain the SmartCS information but also enable the operation of devices connected to the serial port of SmartCS.

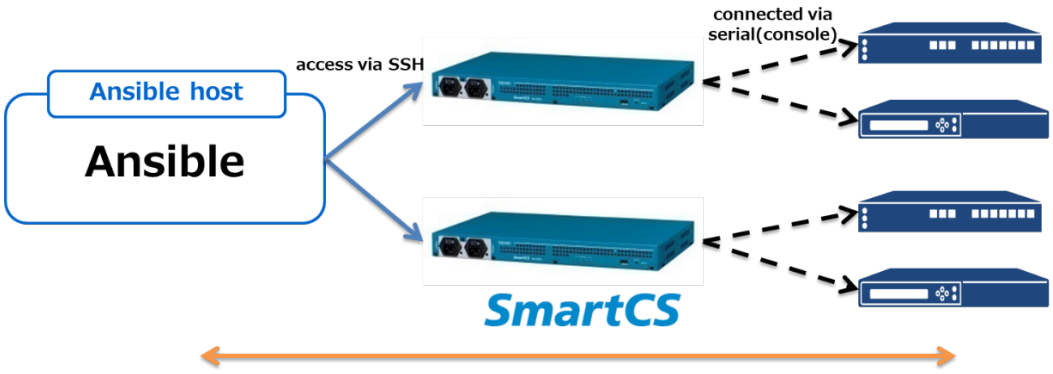
In this document, the function which enables the operation of devices connected to SmartCS is called the "Console Access Function," and the function which configures SmartCS and obtains the SmartCS setting information is called the "CLI Command Function."
Moreover, the Console Access Function can also operate Ansible modules provided by network device vendors via SmartCS.

A list of the functions and modules provided by SmartCS modules for Ansible is shown in the following table.

Function name	Module name
Console Access Function	seiko.smartcs.smartcs_tty_command
CLI Command Function	seiko.smartcs.smartcs_command
	seiko.smartcs.smartcs_config
	seiko.smartcs.smartcs_facts

1.2.1 Console Access Function Overview

Executing CLI commands after logging in to SmartCS via SSH sends and receives characters to the console ports of the network devices connected to the SmartCS serial ports.



Previously, Telnet or SSH protocol had been used to connect to the SmartCS serial ports to directly execute operations manually. Using Ansible enables to execute operations automatically by creating a Playbook.

This function is supported by the following Ansible modules.

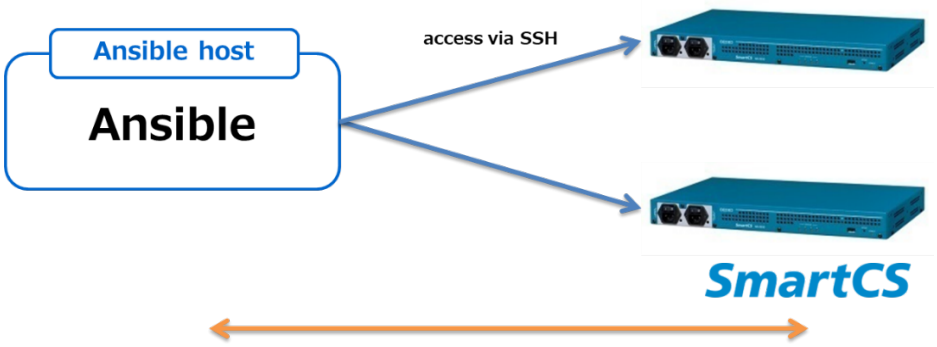
Function name	Module name
Console access function	seiko.smartcs.smartcs_tty_command
	Links the network device vendor module and seiko.smartcs.smartcs_tty_command

Using the smartcs_tty_command module enables you to operate network devices without an Ansible module or IP reachability via Ansible. For the specific usage, refer to "5 Console Access Function."

Moreover, you can also link to modules provided by network device vendors to operate the devices connected to SmartCS. For the specific usage, refer to "6 Linking with Network Device Vendor Modules."

1.2.2 CLI Command Function Overview

This function executes SmartCS CLI commands via Ansible.



This function is supported by the following Ansible modules.

Function name	Module name
CLI command function	seiko.smartcs.smartcs_command seiko.smartcs.smartcs_config seiko.smartcs.smartcs_facts

The summary of the functions provided by each module in the CLI command function is as below.

(1) smartcs_command
Executes any status display command or maintenance command in SmartCS and obtains the results.

(2) smartcs_config
Executes any setting command in SmartCS.

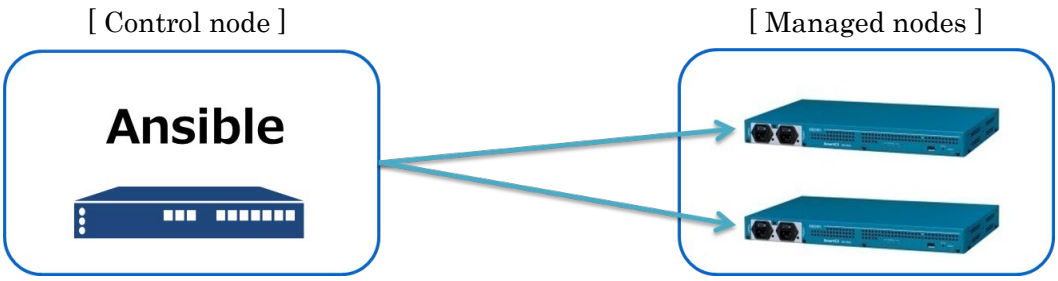
(3) smartcs_facts
Obtains the model name, software version, configuration information, and other device information from SmartCS.

For the specific usage, refer to "7 CLI Command Function."

1.3 Operating environment

1.3.1 Control node / Managed node

The following explains terminology used in this document.
A "control node" is the environment which installs and runs Ansible.
A "managed node" is a SmartCS and must support login via SSHv2 when accessing SmartCS from Ansible.



*When viewed from the perspective of Ansible, which is the control node, a managed node is just a SmartCS.
(They are not devices connected to SmartCS.)

1.3.2 Operating requirements

The operating environments and combinations of the control node Ansible and the managed nodes SmartCS are shown in the table below.

<v1.0 to v1.2>: provided as our original package.

SmartCS modules for Ansible	Control node environment		Managed node environment SmartCS software ver.	
	ansible	Python	NS-2250 series	NS-2240 series
v1.0	2.7.7	2.7 and above 3.6 and above	v2.0 and above	Not supported
v1.1 v1.1.1	2.8.4		v2.1 and above	
v1.2	2.9.15	3.6.8	v2.1 and above	

<v1.3.0 and above>: provided as an Ansible Collections package.

SmartCS modules for Ansible	Control node environment	Managed node environment SmartCS software ver.	
	ansible	NS-2250 series	NS-2240 series
v1.3.0	2.10.x (>=2.10, < 2.11)	v2.1 and above	Not supported
v1.4.0	ansible 2.9.22 and above ansible-base 2.10.x ansible-core 2.11.x (>=2.9.22, < 2.12)		
v1.4.1	ansible 2.9.10 and above ansible-base 2.10.x ansible-core 2.11.x (>=2.9.10, < 2.12)		

Run SmartCS modules for Ansible by combining the control node environment and the managed node environment according to each version.

<Supplemental>

(1) The following type of warning is output when running v1.3.0 and executing a Playbook in an ansible-core2.11 environment.

```
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12.
```

This warning is output because a Python 3.8 environment is required in ansible-core 2.12 and above, but it does not affect the operation.
You can prevent this warning from appearing by adding the following setting to ansible.cfg.

```
deprecation_warnings = False
```

1.3.3 Ansible environment

Ansible can be used by installing it within the Python of the control node environment. The Ansible environment can be built without affecting the Python running on the control node by using the Python virtualization technology venv, so it is recommended that you use venv to build the Ansible environment.

To build an Ansible environment using venv, refer to "11 Appendix A. Building the Ansible Environment."

1.4 License

“SmartCS Modules for Ansible” uses the GNU General Public License Version3 (hereinafter, "GPLv3") as its license. For more details about GPLv3, refer to "15.1 Third-party Software Licenses" and the COPYING file included in the “SmartCS modules for Ansible” package.

For the procedure to create an Ansible Collections package from the source published on GitHub, refer to "15.2 Creating an Ansible Collections Package."

“SmartCS modules for Ansible” includes a modified Ansible program.

1.5 How to get “SmartCS modules for Ansible”

There are three ways to obtain SmartCS modules for Ansible.

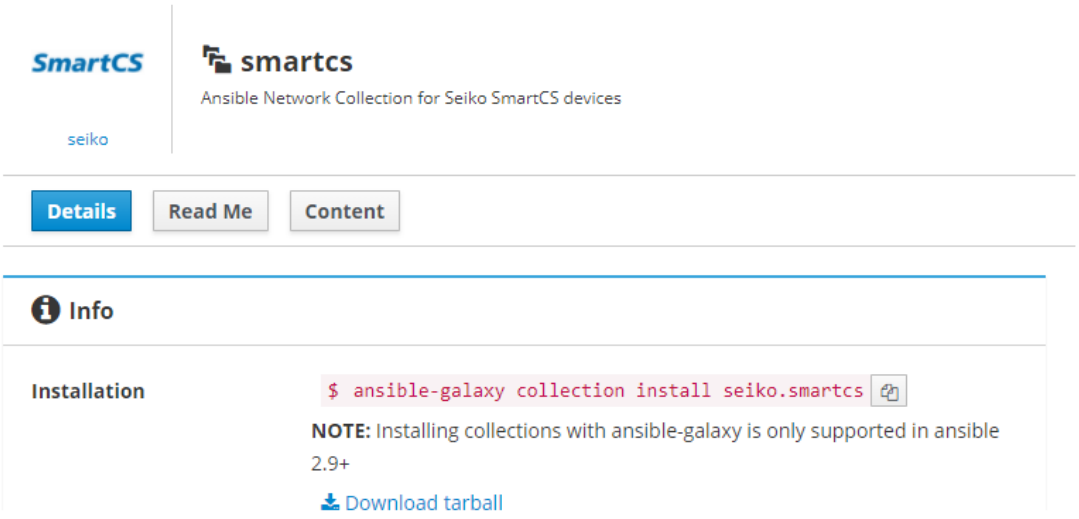
(1) Obtaining SmartCS modules for Ansible from the Ansible Galaxy site

The Ansible Collections package for SmartCS can be obtained from the Ansible Galaxy site.

https://galaxy.ansible.com/

Enter keywords such as "seiko" or "smartcs" to find the Ansible Collections package for SmartCS page.

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



You can download and install it by executing the commands listed under [Details] from the control node.

For the detailed procedure, refer to "2 Installation."

(2) Obtaining SmartCS modules for Ansible from the Ansible Automation Hub site

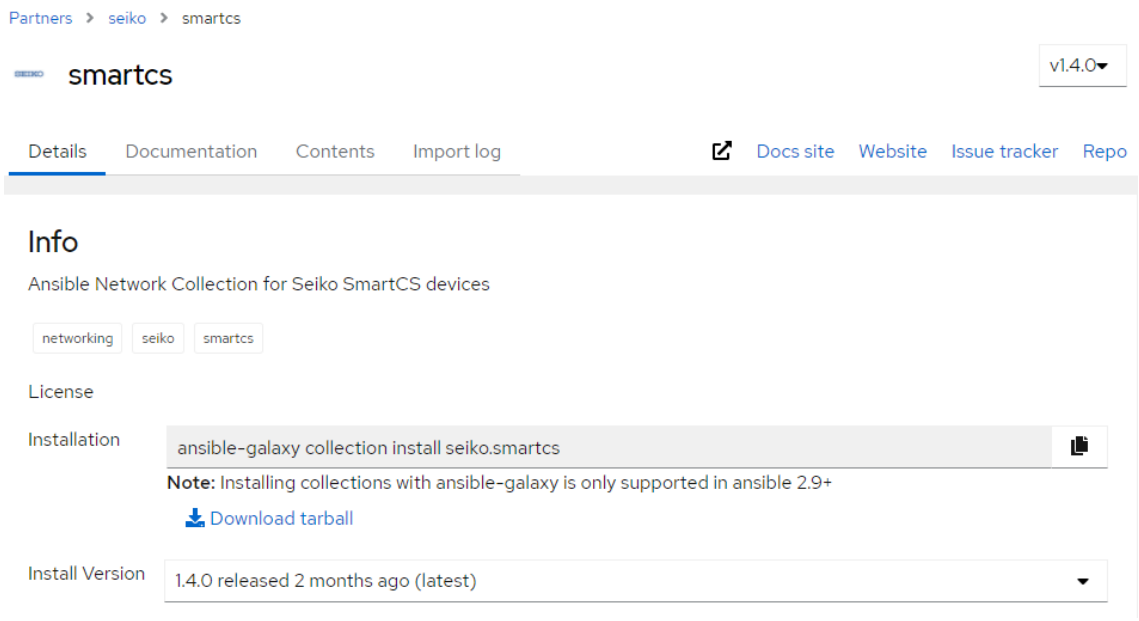
The Ansible Collections package for SmartCS can also be obtained from the Ansible Automation Hub, which is a site that distributes content from Red Hat and authorized partners.

The Ansible Automation Hub is part of the Red Hat Ansible Automation Platform subscription and is only provided to customers with a support contract.

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

Enter keywords such as "seiko" or "smartcs" to find the Ansible Collections package for SmartCS page.

https://console.redhat.com/ansible/automation-hub/repo/published/seiko/smartcs



You can download and install it by executing the commands listed under [Details] from the control node.

For the detailed procedure, refer to "2 Installation."

(3) Obtaining SmartCS modules for Ansible from the SEIKO Solutions web site

The same file which is provided in Ansible Galaxy site is also available from the SEIKO Solutions web site.

If you wish to obtain the software, please apply from the following URL.

https://form.seiko-sol.co.jp/m/dl_orchestrator/

2 Installation

2.1 Pre-installation Check

Check that Ansible is installed. If Ansible is not installed, it’s possible to install it by “yum” or “pip” command, etc. when using CentOS and other operating systems.
For the Ansible environment build method and the ansible.cfg file preparation, refer to "Chapter 11 Appendix A. Building the Ansible Environment."

This chapter explains the procedure for installing modules in the Ansible Collections format which is supported from v1.3.0 of the SmartCS modules for Ansible.

For the procedure to install the version provided as a SEIKO Solutions original package from v1.0 to v1.2, refer to "Chapter 12 Appendix B. v1.0 to v1.2 Operation."

<Provided format and install procedure>

SmartCS modules for Ansible	Provided format	Install procedure
v1.0	SEIKO Solutions original package	"Chapter 12 Appendix B. v1.0 to v1.2 Operation"
v1.1 v1.1.1		
v1.2		
v1.3.0 and above	Ansible Collections format	"Chapter 2 Installation" (This chapter)

2.2 Installation

SmartCS modules for Ansible which supports the Ansible Collections format is provided with the following file name.

File name format	Notes
seiko-smartcs-x.y.z.tar.gz	Ex. v1.3.0 seiko-smartcs-1.3.0.tar.gz

*Regarding the version, there is no particular relationship with the Ansible version.

The version is assigned and provided based on SEIKO Solutions release rules.

Moreover, the namespace and collection name are as follows.

Namespace (namespace name)	seiko
Collection (collection name)	smartcs

This section explains the installation procedures which have been validated by SEIKO Solutions for the SmartCS modules for Ansible in the Ansible Collections format.
When building Ansible in a venv environment, execute each of the steps after transitioning to the venv environment which was built to install the modules.

“SmartCS modules for Ansible” can be installed by one of the following three methods.
There are no differences in the post-installation operation regardless of the install procedure.

(1) Download and install from the Ansible Galaxy site
This is the simplest build procedure. It is recommended that you use this procedure if you are able to access the Ansible Galaxy site from the control node.

(2) Specify the file and install
This procedure directly specifies and installs the Ansible Collections file. Install with this procedure if you obtained the SmartCS modules for Ansible directly from SEIKO Solutions.

(3) Installation using requirements.yml
This install procedure uses the requirements.yml file. By specifying the version of each module and where it can be obtained within the file, you can fix the environment of each package used for the build.

(1) Download and install from the Ansible Galaxy site

SmartCS modules for Ansible are available on the Ansible Galaxy site.
https://galaxy.ansible.com/
https://galaxy.ansible.com/seiko/smartcs

For an ansible 2.10 and above environment, use the ansible-galaxy command to install the SmartCS modules for Ansible.

```
$ ansible-galaxy collection install seiko.smartcs
$
```

Check the SmartCS modules for Ansible version which was installed.

```
$ ansible-galaxy collection list

# /home/test/xxx/xxx/ansible_collections
Collection       Version
---------- -------
seiko.smartcs 1.3.0
$
```

(2) Specify the file name and install

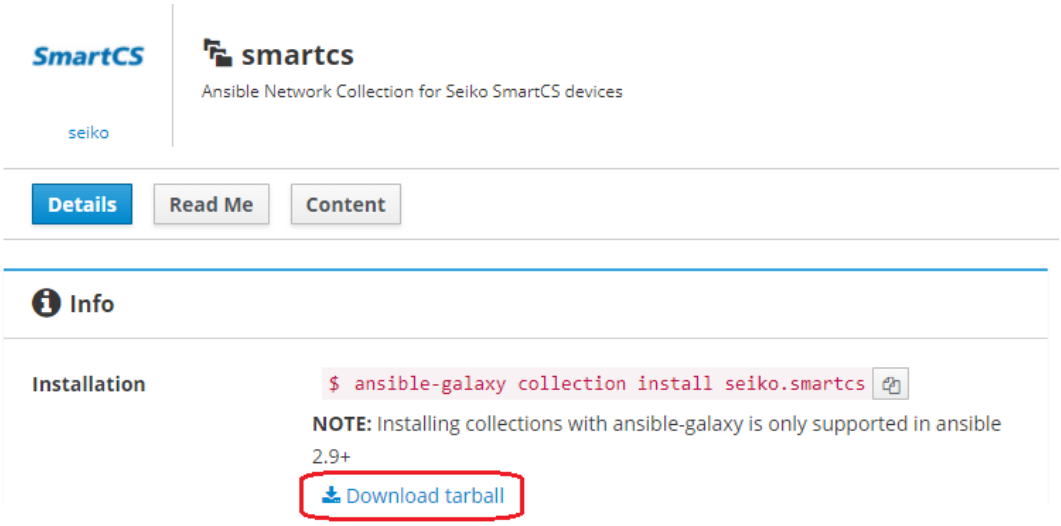
The Ansible Collections package for SmartCS file can be downloaded and obtained as follows.

(i) Use the ansible-galaxy command to download from the Ansible Galaxy site.

```
$ ansible-galaxy collection download seiko.smartcs
$ ls
$ seiko-smartcs-1.3.0.tar.gz
```

*The download destination is the directory specified by collections_paths in ansible.cfg.

(ii) Download from the Ansible Galaxy site using a web browser



(iii) Obtain the package file from SEIKO Solutions.

https://form.seiko-sol.co.jp/m/dl_orchestrator/

After copying the downloaded Ansible Collections package for SmartCS package to the control node, use the ansible-galaxy command to install it.

```
$ ansible-galaxy collection install seiko-smartcs-1.3.0.tar.gz
$
```

Check the SmartCS modules for Ansible version which was installed.

```
$ ansible-galaxy collection list

# /home/test/xxx/xxx/ansible_collections
Collection       Version
---------- -------
seiko.smartcs 1.3.0
$
```

(3) Installation using requirements.yml

You can install modules using the requirements.yml file.

```
---
collections:
  - name: seiko.smartcs
    version: 1.3.0
```

(requirements.yml example)

Because multiple modules can be listed in requirements.yml, this is a convenient mechanism when managing an environment using Ansible Collections.

Specify requirement.yml and execute the ansible-galaxy command to install.

```
$ ansible-galaxy collection install –r requirement.yml
$
```

Check the SmartCS modules for Ansible version which was installed.

```
$ ansible-galaxy collection list

# /home/test/xxx/xxx/ansible_collections
Collection       Version
---------- -------
seiko.smartcs 1.3.0
$
```

2.3 Upgrading

When using the ansible-galaxy command to upgrade the version of an installed Ansible Collections package, you can use either the "--upgrade" option or the "--force" option to upgrade the installed package.

(1) Using the --force option

```
$ ansible-galaxy collection install seiko.smartcs --force
$
```

When the same module is installed and downloaded in an environment where the Ansible Collections package is already installed, it will normally result in an error. However, you can perform an overwrite install by adding the --force option when executing each operation.

This option is also valid for an install using "-r requirements.yml."

```
$ ansible-galaxy collection install –r requirement.yml --force
$
```

(2) Using the --upgrade option (ansible-core2.11 and above)

```
$ ansible-galaxy collection install seiko.smartcs --upgrade
$
```

The --upgrade option was added from ansible-core2.11. You can also use this option to upgrade a module.

2.4 Deleting the installed collection files

With the ansible-galaxy command, a dedicated command for deleting the installed Ansible Collection package is not provided, but it can be deleted by deleting the folder where the installed Ansible Collections files are actually stored.

The directory where the Ansible Collections package is installed can be specified with the collections_paths parameter in ansible.cfg.

https://docs.ansible.com/ansible/latest/reference_appendices/config.html#collections-paths

2.5 About dependent packages

SmartCS modules for Ansible uses network_cli as the connection plugin. Therefore, ansible.netcommon is configured as a dependent package.

If the ansible.netcommon collection, which provides the network_cli connection plugin, is not installed on the control node at the time of installation (when executing the ansible-galaxy collection install command), the process to install the ansible.netcommon collection is also carried out.

2.6 Other

The following explains how to manage the Ansible Collections package using the ansible-galaxy command.

(1) Specify and install (download) a particular version.

When using the ansible-galaxy command to specify seiko.smartcs, the namespace, and collection name to install and download, typically the latest version registered in Ansible Galaxy is automatically installed.

If you wish to specify a particular version, you can install and download it by specifying ":" (colon) and the version name after the collection name.

```
$ ansible-galaxy collection install seiko.smartcs:1.4.0
$
```

Moreover, you can also install and download packages that are assigned identifiers other than the version name with the same procedure.

```
$ ansible-galaxy collection install seiko.smartcs:1.4.0-dev1
$
```

To specify the package with requirements.yml, specify the version as follows.

```
---
collections:
  - name: seiko.smartcs
    version: 1.3.0
```

3 Preparations

3.1 Preparing the SmartCS

The following describes the SmartCS settings required when using either the console access function or the CLI command function.

(1) Terminal output control settings

Configure the terminal output control settings as follows to run the various modules provided by “SmartCS modules for Ansible” in the proper manner.

```
(0)NS-2250# set terminal default prompt device on
(0)NS-2250# set terminal default prompt hostname on
```

*The settings shown above are the default values for the terminal output control settings.

3.2 Preparation for control node

The following describes the preparation on the control node side to execute Ansible which is required when using either the Console Access Function or the CLI Command Function.

(1) Registering the SSH Host Public Key

Before running Playbook, log in to SmartCS via SSH in advance from the management host and register the SmartCS public host key on the management host.

```
[testuser@ansible-host ~]$ ssh smartcs
The authenticity of host 'smartcs (172.31.8.16)' can't be established.
ECDSA key fingerprint is SHA256:/DieiZVP5ggJlupmTPqj/djKRfVRhmhzBPLHZ20jNZ8.
ECDSA key fingerprint is MD5:98:ea:d9:8b:aa:bd:af:13:56:7c:62:ee:7c:6c:d7:61.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'smartcs,172.31.8.16' (ECDSA) to the list of known hosts.
Console Server Authentication.
```

Alternatively, disable the host key check in the ansible.cfg settings (The following "host_key_checking" line is commented out by default to enable host key authentication. When the line is commented out, host key authentication is not performed during SSH connections) when accessing a network device whose host key is not registered on the management host.

```
# uncomment this to disable SSH key host checking
host_key_checking = False
```

4 Creating a Playbook Compatible with Ansible Collections

4.1 Module specification

In Ansible2.10 and above versions standard equipped with the Ansible Collections mechanism, the method for specifying the module name in Playbook creation has changed.
The following explains how to create a Playbook that is compatible with Ansible Collections.

(1) Conventional specification (v1.0 to v1.2)

The module in the tasks section was specified as follows for “SmartCS modules for Ansible” v1.0 to v1.2.

```
tasks:
- smartcs_command:
      commands:
      - show version
      - show config running
```

(2) Specifying in FQCN format

In “SmartCS modules for Ansible” v1.3.0 and above versions that are compatible with the Ansible Collections mechanism, the module name must be specified in Full Qualified Collection Name format (hereinafter "FQCN").

Because the Namespace is "seiko" and the Collection is "smartcs" for SmartCS modules for Ansible, it is specified as "seiko.smartcs.module name."
The following is a Playbook example (task section only) which specifies the smartcs_command module.

```
tasks:
- seiko.smartcs.smartcs_command:
      commands:
      - show version
      - show config running
```

Specify the modules provided by SmartCS modules for Ansible in FQCN format as shown below.

Specification for v1.0 to v1.2	Specification for v1.3.0 and above (FQCN format) *Ansible Collections format specification
smartcs_tty_command	seiko.smartcs.smartcs_tty_command
smartcs_command	seiko.smartcs.smartcs_command
smartcs_config	seiko.smartcs.smartcs_config
smartcs_facts	seiko.smartcs.smartcs_facts

In this document, each module name is basically listed in FQCN format.
*Please note that some locations show images, etc. which show the name specification used for v1.0 to v1.2.

(3) Specifying in the collections directive format

Conventional (v1.0 to v1.2) module name and alias specifications are enabled by using the collections directive within a Playbook.
The specifications for modules provided by SmartCS modules for Ansible with the collections directive are shown below.

```
- name: "Collect the default facts "
  hosts: smartcs

  collections:
  - seiko.smartcs

  tasks:
  - name: "run smartcs_facts with default"
    facts:
      gather_subset:
      - default
```

The following table shows the module names that can be specified when using the collections directive.

Specification for v1.0 to v1.2	Module names when specifying the collections directive
smartc_tty_command	seiko.smartcs.smartcs_tty_command smartcs_tty_command tty_command
smartcs_command	seiko.smartcs.smartcs_command smartcs_command command
smartcs_config	seiko.smartcs.smartcs_config smartcs_config config
smartcs_facts	seiko.smartcs.smartcs_facts smartcs_facts facts

4.2 Connection plugin (network_cli) specification

SmartCS modules for Ansible uses network_cli as the connection plugin. In Ansible2.10 and above versions standard-equipped with the Ansible Collections mechanism, the network_cli connection plugin is a function provided by the ansible.netcommon collection, so the connection plugin and the optional values must be specified in FQCN format in Playbook creation.
Reference: https://galaxy.ansible.com/ansible/netcommon

The method for specifying each in FQCN format is shown below.

Specification for v1.0 to v1.2 (specification up to ansible2.9)	Specification for v1.3.0 and above (FQCN format) *Ansible Collections (ansible2.10 and above) format specification
ansible_connection: network_cli	ansible_connection: ansible.netcommon.network_cli
ansible_network_os: smartcs	ansible_network_os: seiko.smartcs.smartcs
ansible_become_method: enable	ansible_become_method: ansible.netcommon.enable

A Playbook example is shown below.

```
- name: "run show version on SmartCS "
  hosts: smartcs

  tasks:
  - name: "run smartcs_command"
    seiko.smartcs.smartcs_command:
      commands: show version

  vars:
  - ansible_connection: ansible.netcommon.network_cli
  - ansible_network_os: seiko.smartcs.smartcs
  - ansible_user: testuser01
  - ansible_password: testpassword01
  - ansible_become_method: ansible.netcommon.enable
  - ansible_become: yes
  - ansible_become_password: "\n"
```

5 Console Access Function

5.1 Preparing the SmartCS

This section explains the Console Access Function using the smartcs_tty_command module.
For an explanation of linking with the Ansible modules of network device vendors to operate devices connected to SmartCS, refer to "Chapter 6 Linking with Network Device Vendor Modules."

(1) Check the version

Check that the SmartCS system software is v2.1 or above.

```
(0)NS-2250# show version
System                    : System Software Ver 2.1 (Build 2019-MM-DD)
:

(0)NS-2250#
```

(2) Enable SSH connections

Check that the SSH server is enabled.
Change the anthentication method to password authentication from default setting of public key authentication.

```
(0)NS-2250# enable sshd
(0)NS-2250# set sshd auth basic
(0)NS-2250#
```

If the filter function is enabled or the connection permission setting from a specific host is set, set the SSH connection from the management host PC to be allowed.

(3) Create users for extusr groups and granting console access privileges

To use the console access function, a user that belongs to the extusr group must be created. Then configure the user to grant tty manage function and port access.

```
(0)NS-2250# create user smartcs-ansible group extusr password
Changing password for user smartcs-ansible.
New password:
Retype new password:
Password for smartcs-ansible changed
(0)NS-2250# set user smartcs-ansible permission ttymanage on
(0)NS-2250# set user smartcs-ansible port 1-48
(0)NS-2250#
```

(4) Enable the console access function

Configure the setting to enable the console access function.

```
(0)NS-2250# enable ttymanage
(0)NS-2250#
```

The SmartCS setting in order to use the smartcs_tty_command module is shown above.

5.2 Preparation for creating a Playbook

This section explains the setting values required when Playbook creation to use the Console Access Function with the smartcs_tty_command module.

(1) Module setting

Specify "seiko.smartcs.smartcs_tty_command" as the module to use.

(2) Connection plugin setting

Configure "ansible.netcommon.network_cli" as the connection plugin

```
ansible_connection: ansible.netcommon.network_cli
```

(3) Network OS setting

Configure ”seiko.smartcs.smartcs” as the network OS.

```
ansible_network_os: seiko.smartcs.smartcs
```

(4) User name and password settings to connect to SmartCS

Specify the user name and password to connect to SmartCS.

Configure the user who belongs to the extusr group created in section 5-1.

```
ansible_user: smartcs-ansible
ansible_password: password
```

5.3 Playbook Example

```
---
- hosts: smartcs
  gather_facts: no

  tasks:
  - name: "smartcs_tty_command"
    seiko.smartcs.smartcs_tty_command:
      tty: 1
      sendchar :
      - 'show version'

  vars:
  - ansible_connection: ansible.netcommon.network_cli
  - ansible_network_os: seiko.smartcs.smartcs
  - ansible_user: smartcs-ansible
  - ansible_password: password
```

For a description of each options of “smartcs_tty_command” modules, see section 8-1.

6 Linking with Network Device Vendor Modules

6.1 Overview

By enabling the SmartCS SSH transparent connection function (sshxpt), you can operate Ansible modules provided by network device vendors through SmartCS and configure or acquire information from devices connected to SmartCS.

Because the network devices are operated from the console, they can be operated via SmartCS from Ansible even under the default factory settings or if the IP address has not been configured. The control commands described in the Playbook and return value error decisions, etc. operate as defined in the Ansible module which is being used.

To learn more about how to use modules when linking to SmartCS, refer to the documents and web sites provided by each vendor.

6.2 Preparation for SmartCS

The following settings must be configured in SmartCS when linking to network device vendor modules to operate the devices connected to SmartCS.
Steps (2), (3), and (4) are the settings for using the smartcs_tty_command module, and steps (5), (6), and (7) are the settings for linking to and operating network device vendor modules.

(1) Check the version

Check that the SmartCS system software is v2.1 or above.

```
(0)NS-2250# show version
System                    : System Software Ver 2.1 (Build 2019-MM-DD)
:

(0)NS-2250#
```

(2) Enable SSH connections

Check that the SSH server is enabled.

Change the anthentication method to password authentication from default setting of public key authentication.

```
(0)NS-2250# enable sshd
(0)NS-2250# set sshd auth basic
(0)NS-2250#
```

If the filter function is enabled or the connection permission setting from a specific host is set, set the SSH connection from the management host PC to be allowed.

(3) Create users for extusr groups and granting console access privileges

To use the console access function, a user that belongs to the extusr group must be created. Then configure the user to grant tty manage function and port access.

```
(0)NS-2250# create user smartcs-ansible group extusr password
Changing password for user smartcs-ansible.
New password:
Retype new password:
Password for smartcs-ansible changed
(0)NS-2250# set user smartcs-ansible permission ttymanage on
(0)NS-2250# set user smartcs-ansible port 1-48
(0)NS-2250#
```

(4) Enable the console access function

Configure the setting to enable the console access function.

```
(0)NS-2250# enable ttymanage
(0)NS-2250#
```

(5) Creating users for portusr groups and granting console access privileges

To use the SSH Transparent Connection function (sshxpt), create a user in the portusr group and configure it to grant port access privileges.

```
(0)NS-2250# create user smartcs-port group portusr password
Changing password for user smartcs-port.
New password:
Retype new password:
Password for smarcs-port changed
(0)NS-2250# set user smartcs-port port 1-48
(0)NS-2250#
```

(6) Opening port

Configure the sshxpt option for the serial port to enable the SSH transparent connection function (sshxpt) and open the TCP port. The initial value of the port starting number is 9301, and the numbers continue in sequence from the starting number up to the total number of serial ports.

These port numbers correspond to the numbers specified in the "ansible_port" when accessing from Ansible.

```
(0)NS-2250# set portd tty 1 session both both sshxpt
(0)NS-2250#
```

The port starting number can be changed in the settings.

The configurable setting range is from 1025 to 65000.

```
(0)NS-2250# set portd sshxpt 9301
(0)NS-2250#
```

When the filter function is enabled or when permitting access from a specific host, configure SmartCS to allow SSH connections from the management host PC to the port server.

(7) Line feed code setting

Configure SmartCS to send line feed codes when using the SSH transparent connection function (sshxpt) to connect to a network devices. The network devices receives a prompt when the line feed codes are sent, and the various commands described in the Playbook are subsequently executed.

Line feed codes can be selected from CR, LF, CRLF, and none (no transmission) . SmartCS is configured by default to not send line feed codes.

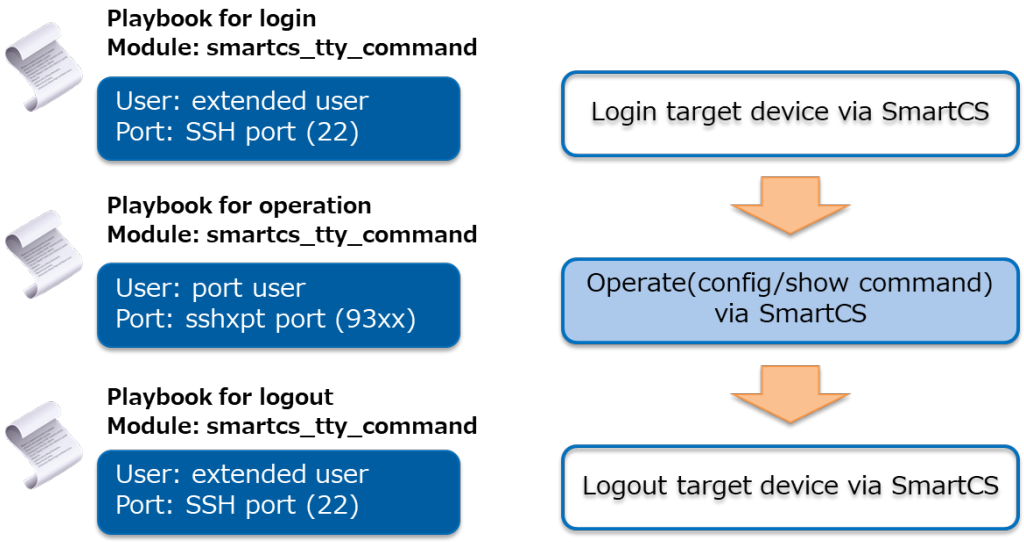
```
(0)NS-2250# set portd tty 1 connted send_nl cr
(0)NS-2250#
```

The SmartCS setting for linking to and operating network device vendor modules is shown above.

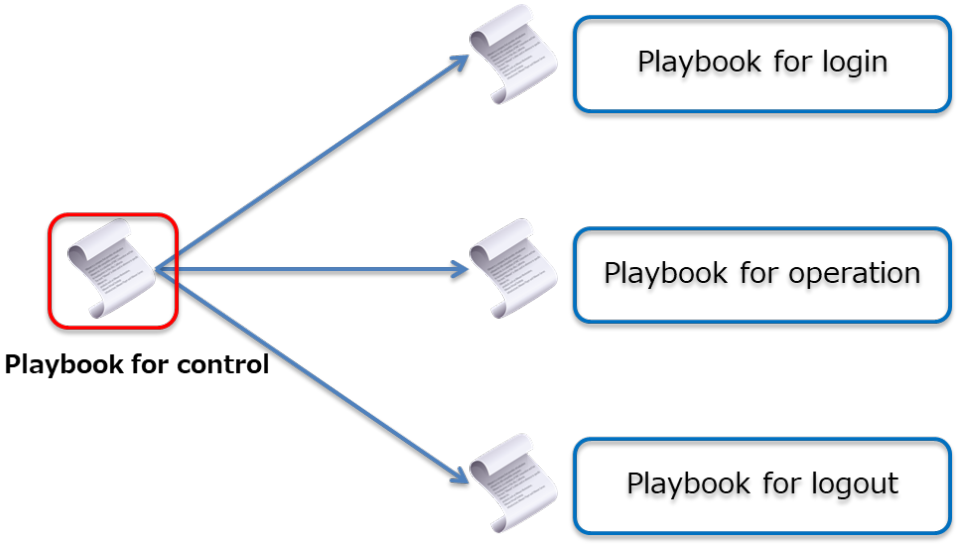
6.3 Preparation for creating a Playbook

It is expected that Ansible modules which use the network_cli connection plugin provided by network device vendors are made so as to normally access via SSH.

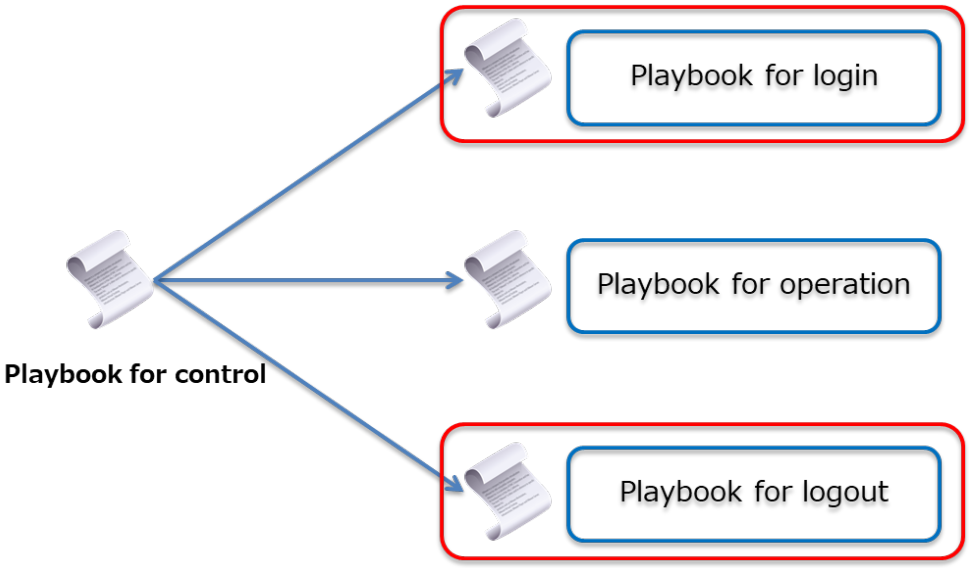
Therefore, in order to achieve a link with SmartCS, a separate Playbook must be created for network device control login and logout processes.

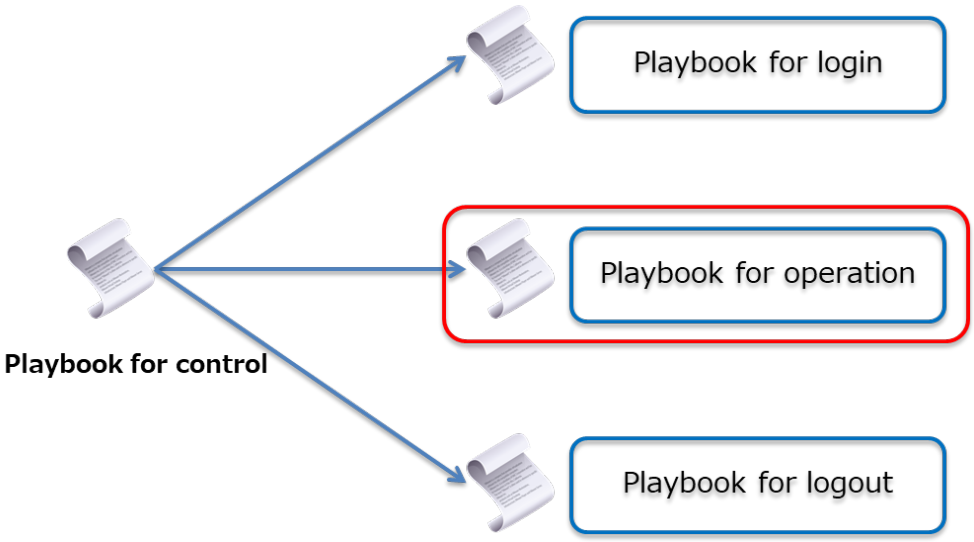


In addition, you can execute a login Playbook, logout Playbook, and operation Playbook as a series of operations by creating a management Playbook as shown in the figure below.



Specify the smartcs_tty_command module in login Playbooks and logout Playbooks. Specify the prompts received in the login and logout processing in recvchar as well as the user name, password, and logout commands, etc. for the sendchar of each Playbook to create a Playbook to enalbe logging in to andout from the network devices.





In an operation Playbook, specify the third-party module to link to and describe the commands that you wish to execute. Use the TCP port opened when enabling the SSH transparent connection function (sshxpt) to connect to the network devices. (Under the default setting, tty1 operations use port 9301.)

The following explains the setting values required to create a Playbook for operation.

(1) Module

Specify the network device vendor module to link.

(Ex.)

"xxx_command"

"xxx_config"

"xxx_facts"

(2) Connection plugin setting

Configure "ansible.netcommon.network_cli" as the connection plugin.

```
ansible_connection: ansible.netcommon.network_cli
```

(3) Network OS

Specify the network OS to specify in the network device vendor module to be collaborated.

```
ansible_network_os: xxx
```

(4) User name and password settings to connect to SmartCS

Specify user name and password of the SmartCS port user group created in "6.2 (2) Grant user creation and console access authority to the port user group."

(5) Port number

Specify the connection port configured in "6.2 (3) Open the connection port."

```
ansible_port: 9301
```

6.4 Playbook example

Ex.) Management Playbook

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

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

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

Ex.) Login Playbook (login.yml)

```
---
- hosts: smartcs
   tasks:
   - name: "Login by Console"
      seiko.smartcs.smartcs_tty_command:
         tty: 1
         recvchar:
         - 'username: '
         - 'password: '
         - 'switch>'
         sendchar:
         - '__NL__'
         - 'user'
         - 'secret'

   vars:
   - ansible_connection: ansible.netcommon.network_cli
   - ansible_network_os: seiko.smartcs.smartcs
   - ansible_user: smartcs-ansible
   - ansible_password: password
```

Ex.) Logout Playbook (logout.yml)

```
---
- hosts: smartcs
   tasks:
   - name: "Logout by Console"
      seiko.smartcs.smartcs_tty_command:
         tty: 1
         recvchar:
         - 'username: '
         - 'password: '
         - 'switch>'
         sendchar:
         - 'exit'

   vars:
   - ansible_connection: ansible.netcommon.network_cli
   - ansible_network_os: seiko.smartcs.smartcs
   - ansible_user: smartcs-ansible
   - ansible_password: password
```

For a descripition of the each “smartcs_tty_command” module options, see section 8-1-2.

Ex.) Operation Playbook (operation.yml)

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

   vars:
   - ansible_connection: ansible.netcommon.network_cli
   - ansible_user: smartcs-port
   - ansible_password: password
   - ansible_port: 9301
   - ansible_network_os: xxx.xxx.xxx
   - ansible_become: yes
   - ansible_become_method: ansible.netcommon.enable
   - ansible_become_password: secret
   - ansible_command_timeout: 60
```

6.5 Instructions and Directions for Use

The limitations and precautions when linking to third-party modules to operate network devices are described below.

(1) network_cli connection plugin support
Only third-party modules which support “network_cli” as a connection plugin can link to SmartCS.

(2) Network device prompt specification
The network devices prompt definition must be the same when accessing via SSH or from the console.

(3) Processing speed care (timeout period adjustment)
Control is performed from the console of the network devices, so the processing speed is slower than connecting directly to the device via SSH or a regular Ansible operation. Therefore, the timeout specified by "ansible_command_timeout" must be set to a longer period.

7 CLI Command Function

7.1 Preparing the SmartCS

The following describes the SmartCS preparations required to use the CLI command function.

(1) Enable SSH connections

Check that the SSH server is enabled.

Change the authentication method to password authentication from default setting of public key authentication.

```
(0)NS-2250# enable sshd
(0)NS-2250# set sshd auth basic
(0)NS-2250#
```

If the filter function is enabled or the connection permission setting from a specific host is set, set the SSH connection from the management host PC to be allowed.

7.2 Preparations for creating Playbook

The following explains the setting values required to create a Playbook when executing the CLI Command Function.

(1) Module setting

Specify the modules to use from the following options.

”smartcs_command”

“smartcs_config”

“smartcs_facts”

(2) Connection plugin setting

Configure "ansible.netcommon.network_cli" as the connection plugin.

```
ansible_connection: ansible.netcommon.network_cli
```

(3) Network OS setting

Configure "seiko.smartcs.smartcs" as the network OS.

```
ansible_network_os: seiko.smartcs.smartcs
```

(4) User name and password settings to connect to SmartCS

Specify the user name and password to connect to SmartCS.

This setting works with users created in either of the following groups.

・General user group

・Extusrr group

(5) Privileged user setting

Depending on the module or options used, the module does not function correctly without switching to the privileged user.

In such a case, configure the Playbook as follows.

```
ansible_become: yes
ansible_become_method: ansible.netcommon.enable
ansible_become_password: password (privileged user password)
```

7.3 Module and privileged user

The following table lists modules and options which do not function correctly without switching to the privileged user.

○: Requires a transition to the privileged user

-: Operates normally with or without trannsition to the privileged user.

Module name	Option name	Transition to the privileged user
smartcs_command	commands	Depends on the commands specified by the “commands” option.
smartcs_config	lines	○
	src	○
smartcs_facts	all	○
	default	—
	tty	—
	config	○

Refer to the Command Reference for information on whether you need to transition to a privileged user for optional commands.

The “smartcs_facts” option name is the value specified by the “gather_subset” option. Also, for example "!tty" is specified, changing to privileged user is required since everything other than the tty option is specified.

7.4 Playbook Example

```
---
- name: smartcs_command
  hosts: smartcs
  gather_facts: no
  tasks:
  - seiko.smartcs.smartcs_command:
        commands:
        - show version
        - show config running

  vars:
  - ansible_connection: ansible.netcommon.network_cli
  - ansible_network_os: seiko.smartcs.smartcs
  - ansible_user: somebody
  - ansible_password: password
  - ansible_become: yes
  - ansible_become_method: ansible.netcommon.enable
  - ansible_become_password: "\r"
```

See section 8-2-2 for a description of each option in the “smartcs_command” module.

If a password is not givent to the privileged user, configure "\r" for “ansible_become_password” as shown in the example above.

8 Modules

8.1 seiko.smartcs.smartcs_tty_command

The following explains the modules used with the console access function.

8.1.1 Overview

This function sends the specified characters to the console port of the network devices connected to the SmartCS serial port and retrieves the console input/output results.

8.1.2 Options

The following table lists the options for this module.

Option name	Required	Default	Setting range	Description
cmd_timeout		10	1 to 7200	Set the timeout (seconds) from sending the characters specified in sendchar/src until receiving the characters specified in recvchar.
error_detect_on_sendchar		cancel	cancel exec	Set whether or not to send the following characters when an error occurs after sending the characters specified in sendchar/src. When set to "cancel," characters are not sent after a character transmission error occurs. When set to "exec," the next characters are sent even after a character transmission error occurs.

Option name	Required	Default	Setting range	Description
error_detect_on_module		ok	ok failed	Set whether to configure the ansible command (ansible-playbook command) execution result to "ok" or "failed" when an error occurs after sending the characters specified in sendchar/src. When set to "ok," the ansible command result is "ok" without resulting in an error even if an error occurs after sending the characters. When set to "failed," the ansible command results in an error and is designated as "failed" when an error occurs after sending the characters.
error_recvchar_regex				Set a list of characters using regular expressions to detect an error when the characters received after sending the characters specified in sendchar/src include a particular set of characters. This setting can be specified in list format with a maximum of eight entries. In the event that an error is detected, the ansible command is set to "ok" or "failed" according to the error_detect_on_module option setting.
nl		cr	crlf cr lf	Set the character line feed code sent by sendchar/src.

Option name	Required	Default	Setting range	Description
recvchar				Set a list of received characters to wait for after sending the characters specified in sendchar/src. This setting can be specified in list format with a maximum of 16 entries.
recvchar_regex				Use regular expressions to set the character list specified in recvchar. This setting can be specified in list format with a maximum of eight entries.

Option name	Required	Default	Setting range	Description
sendchar	(○)			Set the characters to send to the target tty. The transmitted character string is sent in order from the top of the list specified in sendchar. In addition to sending the specified characters, the following sending methods can also be specified for the sendchar option. __NL__ Send only the line feed code. The line feed code is the value set in the nl option. __CTL__:hex Send the control character. For the table of specified hex values and the corresponding control characters, refer to "8.1.5-(4)-5 Sending control characters." __WAIT__:sec Set the timeout time to wait until the characters specified in recvchar are received for each transmitted character string. When left unspecified, the timeout time is set to the value configured in the cmd_timeout option. __NOWAIT__ Do not wait for the characters specified in recvchar for each transmitted character string.

				Immediately send the next characters after sending the characters specified in sendchar/src. __NOWAIT__:sec Do not wait for the characters specified in recvchar for each transmitted character string. After sending the characters specified in sendchar/src, wait for the specified period of time (seconds) before sending the next characters. The range of time that can be configured in the sendchar option is from 1 to 7200 (seconds). It is recommended that the list of characters sent with sendchar be enclosed within single or double quotation marks. Either sendchar or src must be specified. This option runs exclusively from the src option.
src	(○)			Specify the path to the file which lists the characters to send to the target tty line by line. This option specifies either an absolute path of the specified file, or a relative path from the Playbook save directory. Either src or sendchar must be specified. This option runs exclusively from the sendchar option.

Option name	Required	Default	Setting range	Description
tty	○		1 to 48	Set the tty to send the characters to. This option can be configured in ttylist format (1-16, 1, 2-8, 16). Send the characters specified in sendchar/src for each tty number when multiple tty numbers are specified.
ttycmd_debug		off	off on detail	Display the following information after the character transmission processing by means of sendchar/src has ended. ・tty option setting ・cmd_timeout option setting ・nl option setting ・ error_detect_on_sendchar option setting ・recvchar option setting ・recvchar_regex option setting ・error_recvchar_regex option setting This option is used for debugging.

Option name	Required	Default	Setting range	Description
custom_response		False	boolean value	Send a return value in a format which can distinguish between transmitted characters and received characters in addition to stdout and stdout_lines. Separate and output the execute_command and response for each transmitted character string.
custom_response_delete_nl		False	boolean value	Delete only the line feed line for the custom_response output.
custom_response_delete_lastline		False	boolean value	Delete the last line of the response for the custom_response output. *The purpose of this option is to not display the prompt after executing a CLI command.

Option name	Required	Default	Setting range	Description
initial_prompt				Specify the characters which are expected to be received after sending the initial_prompt_check_cmd. This setting can also be specified using a regular expression. The pre-check processing runs when this setting is specified.
initial_prompt_check_cmd		__NL__ (line feed)		Specify a command to check the console prompt status before sending the characters specified in sendchar/src. If left unspecified, it sends __NL__ (line feed).
initial_prompt_check_cmd_timeout		5	1 to 30	Specify the time to wait until checking the received characters after sending the initial_prompt_check_cmd.
escape_cmd				Specify the characters to send when unable to receive the initial_prompt after sending the initial_prompt_check_cmd.
escape_cmd_timeout		5	1 to 30	Specify the time to wait until checking if the initial_prompt is included after sending the escape_cmd.

escape_cmd_retry		3	0 to 8	Specify the number of times to retry sending the initial_prompt_check_cmd when the initial_prompt is not received after sending the escape_cmd.

8.1.3 Playbook Example

The following shows a Playbook example for this module.

```
---
- name: Configure_ipaddress
   hosts:    smartcs
   gather_facts: no

   tasks :
   - name : Configure   NS-2250   ipaddress   by Console
      seiko.smartcs.smartcs_tty_command :
         tty: 1
         nl : cr
         cmd_timeout : 5
         recvchar :
         - 'NS-2250 login: '
         - 'Password: '
         - '(c)NS-2250> '
         - '(c)NS-2250# '
         - '[y/n] ? '
         - 'logout: somebody/console'
         sendchar :
         - '__NL__'
         - 'somebody'
         - '__NL__'
         - 'su'
         - '__NL__'
         - 'set ipaddr eth1 192.168.0.1/24'
         - 'write'
         - 'y'
         - 'exit'
         - 'exit'

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

8.1.4 Return Values

The following table lists the return values for this module.

Name	Description	Trigger	Type
stdout_lines	Output a list separated by each line feed character for the characters sent and received by the console. The return value creates a list for each transmitted character string specified in sendchar/src.	When the command execution is successful	List
stdout	Output the characters sent and received by the console. The return value creates a list for each transmitted character string specified in sendchar/src.	When the command execution is successful	List
pre_stdout_lines	Output a list separated by each line feed character for the characters sent and received by the console when the pre-check function works.	When the initial_prompt setting is configured and the command execution is successful	List
pre_stdout	Output the characters sent and received by the console when the pre-check function works.	When the initial_prompt setting is configured and the command execution is successful	List
stdout_lines_custom	For the characters sent and received by the console, output a list in a format which distinguishes between transmitted characters	When the custom_response setting is enabled and the command execution is successful	List

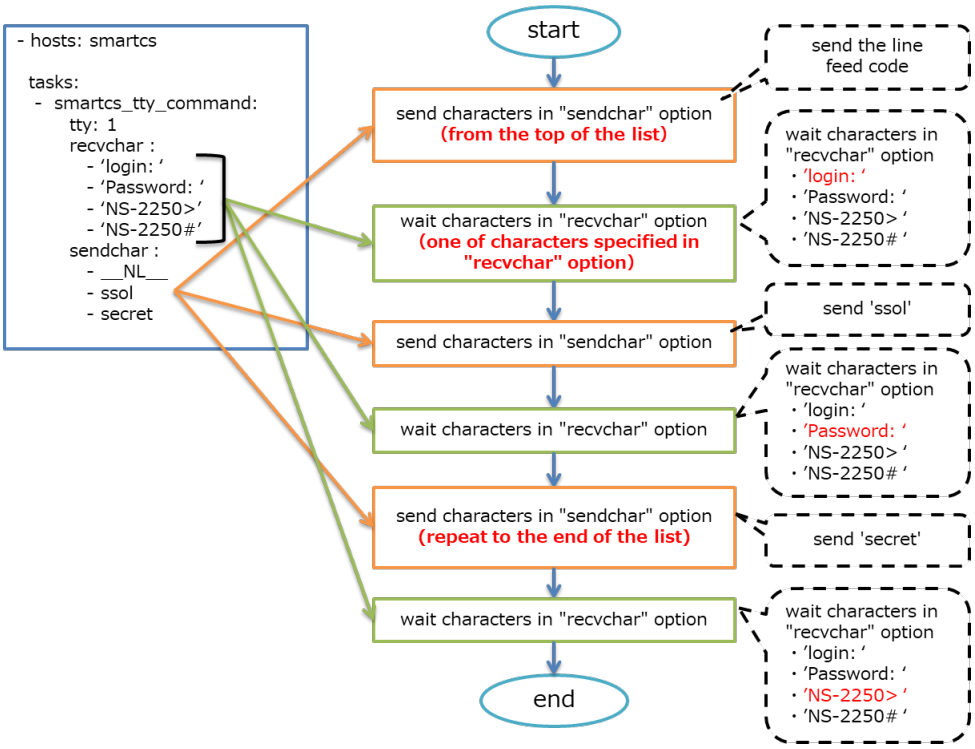
	(execute_command) and received characters (response).		

8.1.5 Explanations

The following explains the operation of each option.

(1) sendchar/src and recvchar operation

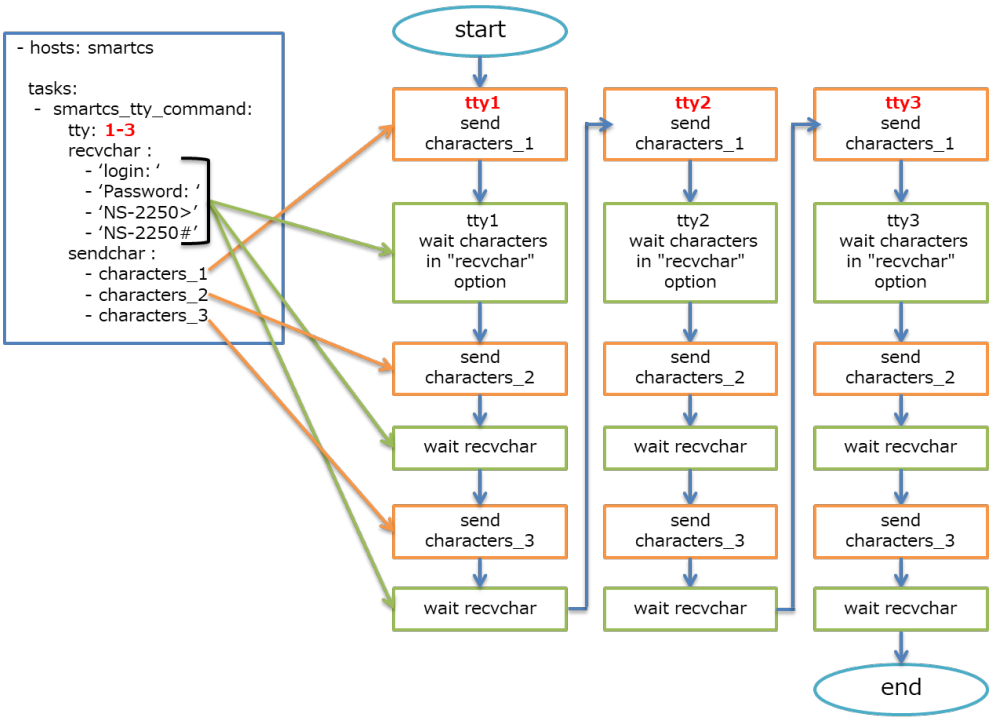
sendchar/src sends the specified characters in order from the top.
recvchar waits to see whether or not the matching characters are included in the input/output content after sending the characters. When the matching characters are received, it sends the next characters.
*The figure below shows an example of specifying the sendchar option.



(2) Operation when specifying multiple tty's

The tty option can be configured in ttylist format (multiple specifications including hyphens and commas). If multiple tty's are specified, it sends the characters specified in sendchar/src for each tty number.

*The figure below shows an example of specifying the sendchar option.

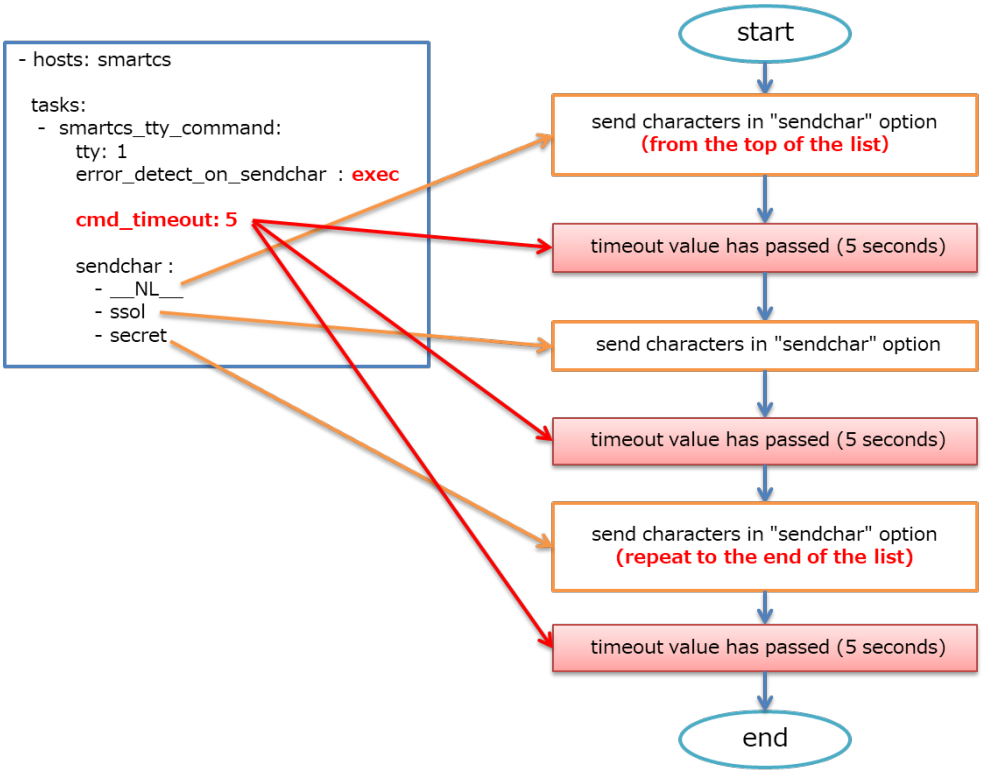


(3) Operation when recvchar is not configured

If recvchar (recvchar_regex) is not configured, sendchar/src waits until the cmd_timeout period has passed before sending the next characters.

*The only parameters which are essential for the smartcs_tty_command module are the tty and sendchar/src options.

*The figure below shows an example of specifying the sendchar option.



If error_detect_on_sendchar is set to "cancel" (default value), it does not send the next characters if an error occurs (timeout error indicated above) when sending the characters specified in sendchar/src. Therefore, error_detect_on_sendchar is set to "exec" in the example above.

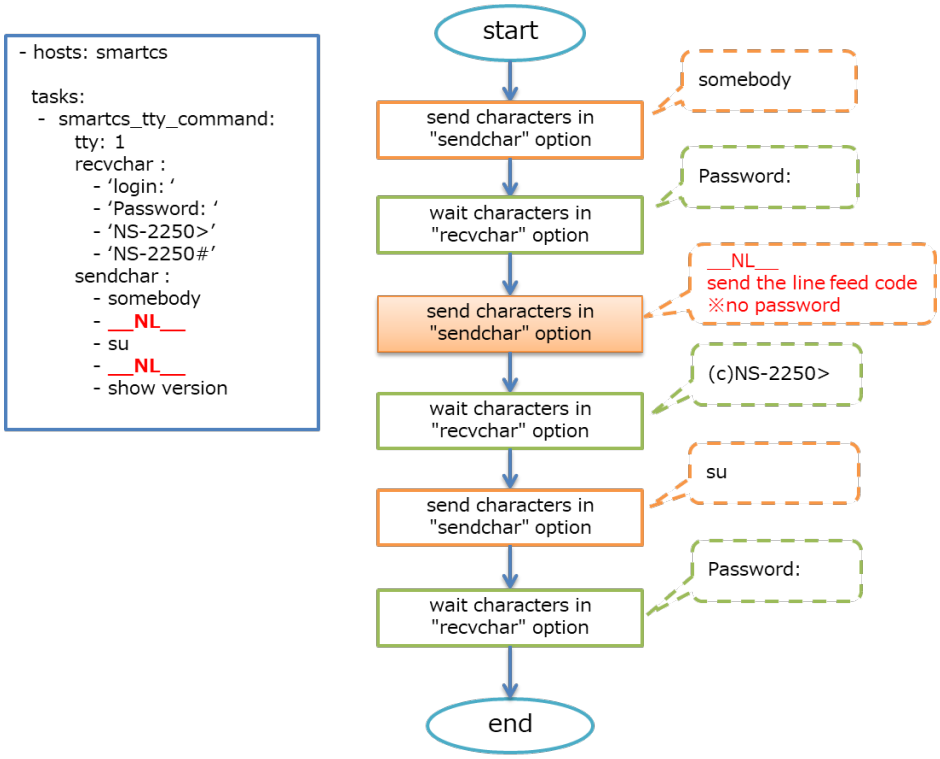
(4) Special sendchar/src settings

In addition to sending the specified characters, sendchar/src can be set to special sending methods by specifying certain options.

1. Send only the line feed code.

__NL__ option

Set the transmitted character string to "__NL__" to send only the line feed character.
The transmitted line feed code is the value set in the nl option.
This option can be used when setting a blank password in password entry situations such as performing login processing operations from the console.
*The figure below shows an example of specifying the sendchar option.



2. Set a timeout period for each transmitted character string.

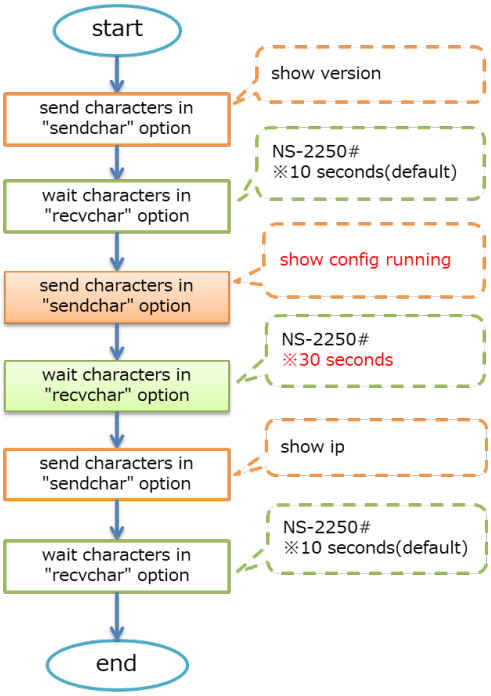
__WAIT__:sec option

The characters specified in sendchar/src wait for the characters configured in recvchar for the period of time set in the cmd_timeout option (default of 10 seconds).
If the commands executed on the network devices console by the transmitted character string take time to output the results (execution of commands to retrieve the running config or support information), only the specific transmitted character string can change the timeout period.

In the following example, the timeout value for recvchar is the default of 10 seconds, but the timeout period is set to 30 seconds only when sending the "show config running" characters.
*The figure below shows an example of specifying the sendchar option.

```
- 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
```

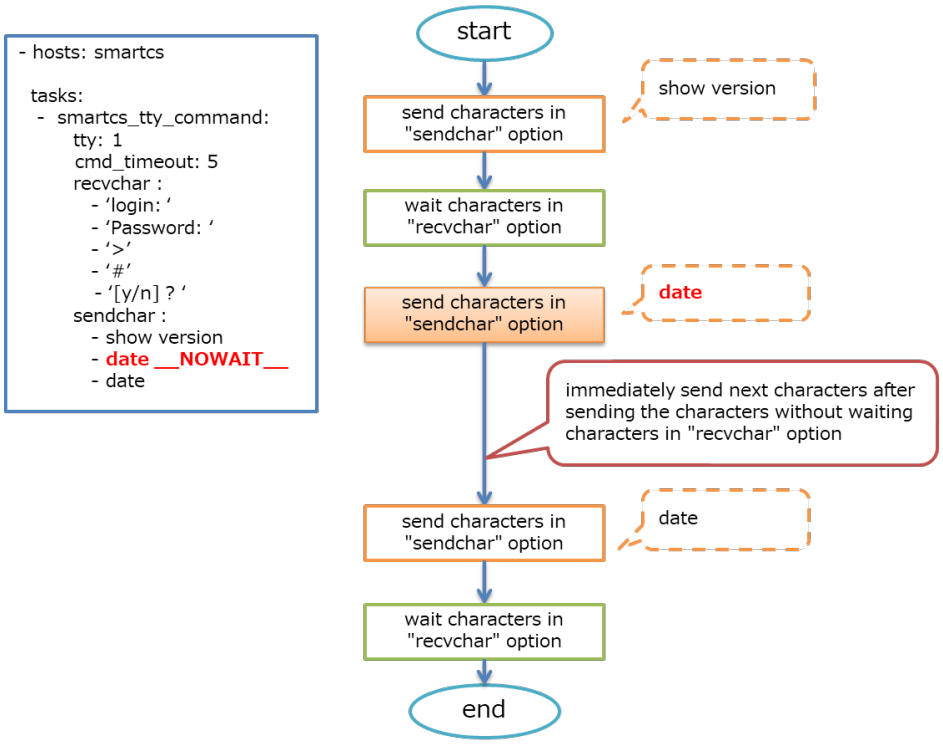


3. Setting to not wait for recvchar for each transmitted character string.

__NOWAIT__ option

This option immediately sends the next characters without waiting for the period of time set in the cmd_timeout option.
*Transmits approximately one second later.

This option can be used when you wish to sequentially send characters to the console connection destination without waiting for recvchar.
*The figure below shows an example of specifying the sendchar option.



4. Setting to wait only for a set period of time without waiting for recvchar for each transmitted character string.

   __NOWAIT__:sec option

   If the recvchar setting is configured, this option checks whether or not the recvchar are included in the input/output results after sending the characters and sends the next characters if they are included.
   However, depending on the operation that you wish to perform on the network devices console, it may not function as intended based on these fundamental operations.

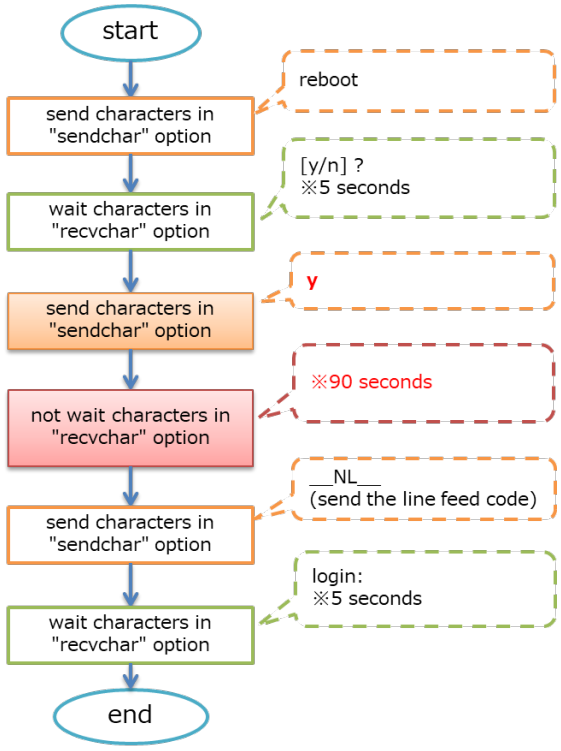
   (Ex.)
   ・Characters such as "#" and ">" are set in recvchar, which would normally wait for a prompt from the network devices, but ">" is included in the output of the executed command, and the next string is sent.

   ・When executing a reboot or a version upgrade command, various characters and symbols are output to the console, which unintentionally matches recvchar, and the next characters are sent during the reboot, etc.

   In order to perform console operations as intended through the Playbook as much as possible in the kinds of situations described above, you can set the option to not wait for recvchar for each transmitted character string.
   *The figure below shows an example of specifying the sendchar option.

```
- 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
```



5. Send a control character

__CTL__ option

Specify "__CTL__:hex" in sendchar/src when sending a control character.
The following range of control characters can be sent.

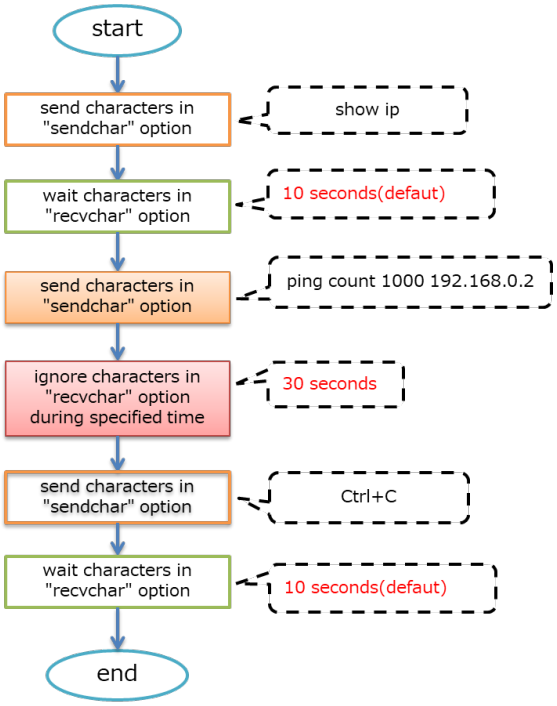
00 : [Ctrl-@]	08 : [Ctrl-H]	10 : [Ctrl-P]	18 : [Ctrl-X]
01 : [Ctrl-A]	09 : [Ctrl-I]	11 : [Ctrl-Q]	19 : [Ctrl-Y]
02 : [Ctrl-B]	0a : [Ctrl-J]	12 : [Ctrl-R]	1a : [Ctrl-Z]
03 : [Ctrl-C]	0b : [Ctrl-K]	13 : [Ctrl-S]	1b : [Ctrl-[ ] / ESC
04 : [Ctrl-D]	0c : [Ctrl-L]	14 : [Ctrl-T]	1c : [Ctrl-\ ]
05 : [Ctrl-E]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

*The leftmost value is the "hex" part of __CTL__:hex and the value specified in the Playbook.

This option can be used when sending a control character from the console.
Ex: stop the ping execution. Send after executing a command on a specific network device, etc. *The figure below shows an example of specifying the sendchar option.

```
- hosts: smartcs

 tasks:
  -  smartcs_tty_command:
       tty: 1
       recvchar :
         - 'Username: '
         - 'Password: '
         - '(c)NS-2250>'
         - '(c)NS-2250#'
       sendchar :
         - show ip
         - ping count 1000 192.168.0.2 __NOWAIT__:30
         - __CTL__:03
```



6. Combine special sending methods

The following table shows combinations of sendchar/src sending methods.

Configuration method	Notes
show version	Send characters
show version __WAIT__:sec	Wait for recvchar for the configured period of time after sending the characters
show version __NOWAIT__	Immediately send the next characters after sending the characters.
show version __NOWAIT__:sec	Wait only for the configured period of time after sending the characters.
__NL__	Send a line feed
__NL__ __WAIT__:sec	Wait for recvchar for the configured period of time after sending a line feed
__NL__ __NOWAIT__	Immediately send the next characters after sending a line feed.
__NL__ __NOWAIT__:sec	Wait only for the configured period of time after sending a line feed.
__CTL__:03	Send a control character.
__CTL__:03 __WAIT__:sec	Wait for recvchar for the configured period of time after sending the control character.
__CTL__:03 __NOWAIT__:sec	Send the next characters after sending the control character and waiting for the configured period of time.
__CTL__:03 __NOWAIT__	Immediately send the next characters after sending the control character.

*The "show version" part is an example of when some kind of transmitted

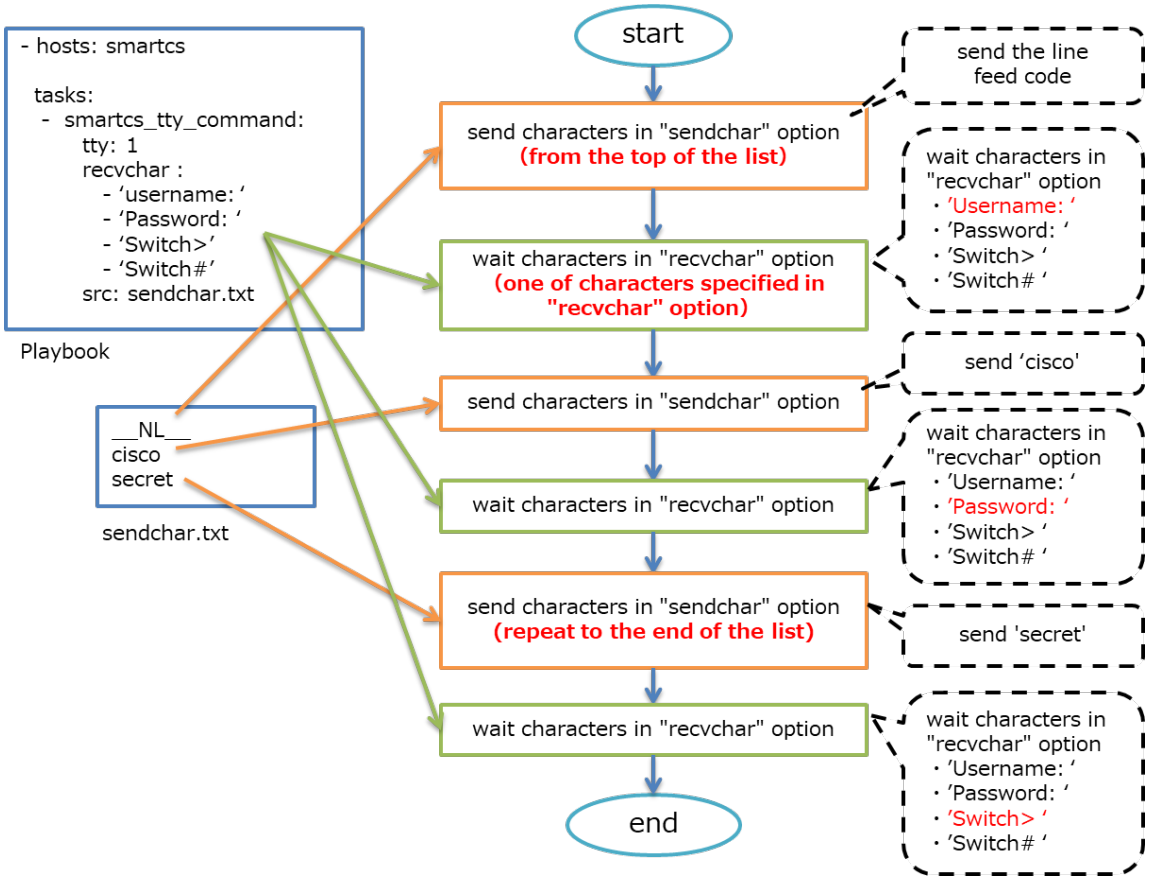
character string is specified.

*The 03 specified in the "__CTL__" setting is an example of specifying Ctrl+C.

(5) Specify the transmitted character string with an external file

In addition to declaring the transmitted character string in a list format with the sendchar option in a Playbook, an external file can also be loaded and executed with the src option.

*The figure below shows an example of specifying the src option.



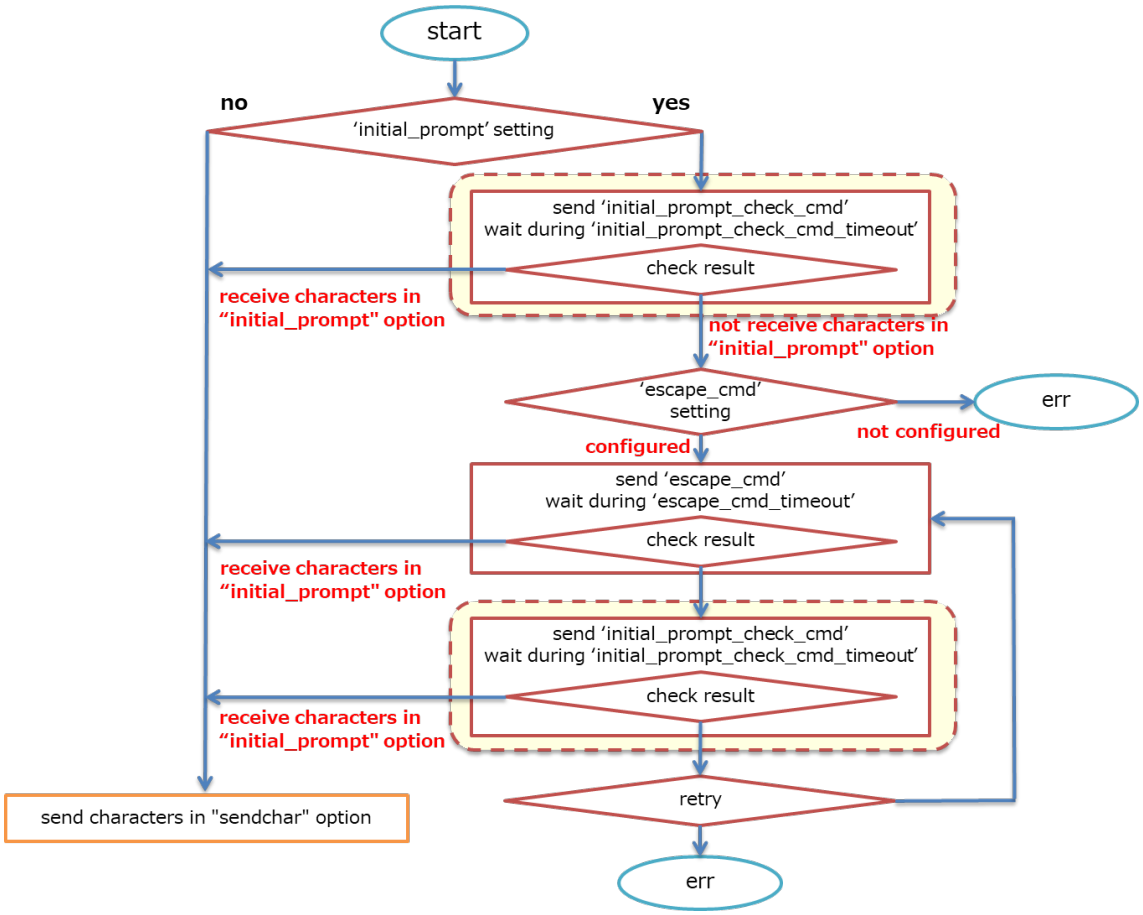
(6) Function for checking the console status before sending the characters (pre-check function)

You can check the state expected by the console of the network devices (ex: login prompt state) before sending the characters specified in the sendchar/src option.

Send any command (default is a line feed) specified in initial_prompt_check_cmd before sending the characters specified in the sendchar/src option by setting the initial_prompt option. Check whether or not there is a partial match between the characters specified in initial_prompt and those included in the command result. If there is no match, send any command specified in escape_cmd (ex: exit) and check the result once again for a partial match.

The initial_prompt can be configured with a regular expression.

Refer to the figure below for the operational flow of the pre-check function.



<Supplemental>

1. This function operates when the initial_prompt option is configured.
   A line feed is sent when initial_prompt_check_cmd is unspecified.
   *By default, __NL__ is configured, and the line feed code depends on the NL option.

2. When this function is running, the Playbook results in an error before sending the characters specified in sendchar/src in the following instances.

   ・escape_cmd setting is not configured
   initial_prompt_check_cmd is sent, and the expected characters are not received.

   ・escape_cmd setting is configured, and the number of retries reaches the upper limit
   The expected characters are not received after sending escape_cmd for the number of retries.

3. When this function runs, and the Playbook exits normally, the following return values are also output.
   The values for the console input/output results are output when sending initial_prompt_check_cmd and escape_cmd.

   ・pre_stdout_lines
   Returns a list separated by each line feed character of the console input/output content sent and received in the check processing before sending the characters specified in sendchar/src.

   ・pre_stdout
   Returns the console input/output content sent and received in the check processing before sending the characters specified in sendchar/src.

4. Even if this function is used, there is no guarantee that it will return the status expected by the console in every situation.

(7) error_detect_on_sendchar operation

In some cases, sending the characters specified in sendchar/src may result in an error for the following reasons.

<Causes of character transmission errors>

Error cause		Cause
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 configured in error_recvchar_regex.	Error:: Matched “xxx”.
Do not send the next characters when error_detect_on_sendchar is set to "cancel"		Error:: After error.

If the next characters are sent when these errors occur, the system may operate in a different way than expected.
For this reason, error_detect_on_sendchar is provided as an option to configure the following operations

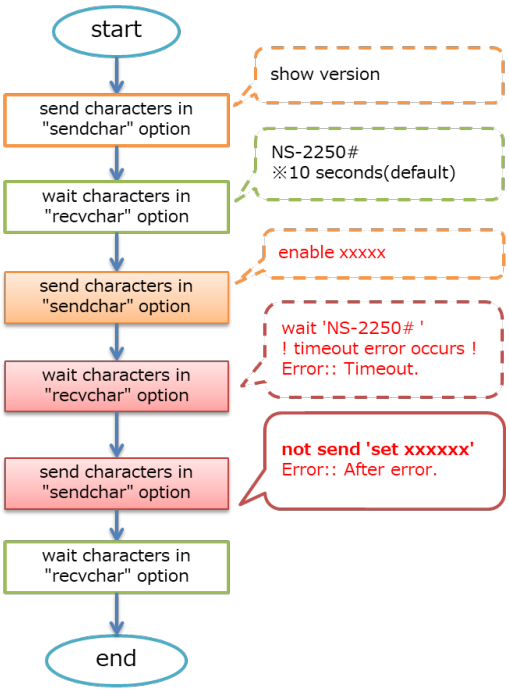
・Send the characters immediately after the error occurs

・Do not send the characters immediately after the error occurs

1. Operation when error_detect_on_sendchar:cancel is configured

```
- 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
```

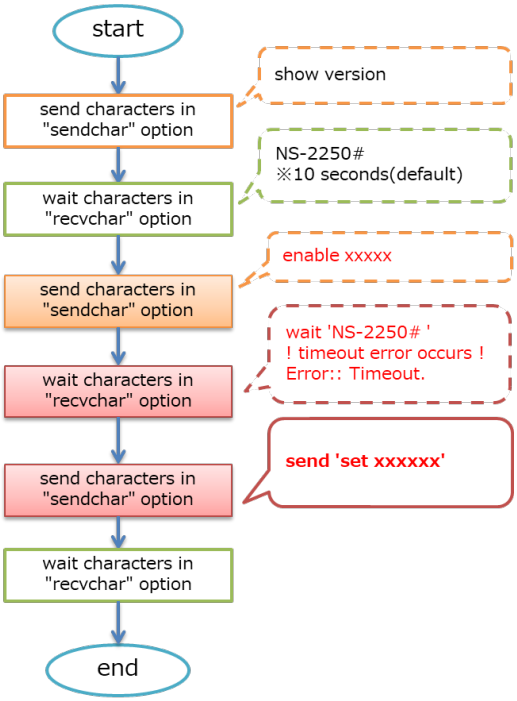


*The default value is error_detect_on_sendchar:cancel.

2. Operation when error_detect_on_sendchar:exec is configured

```
- 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
```



(8) error_detect_on_module operation

In the smartcs_tty_command module, it directly returns the console input/output, so it fundamentally returns "ok" for the ansible command execution result.
However, the error within the smartcs_tty_command module can be controlled and the ansible command set to "failed" by using the error_detect_on_module option. The errors which can be controlled are limited to the following "errors occurring when sending characters configured with the sendchar option in the smartcs_tty_command module."

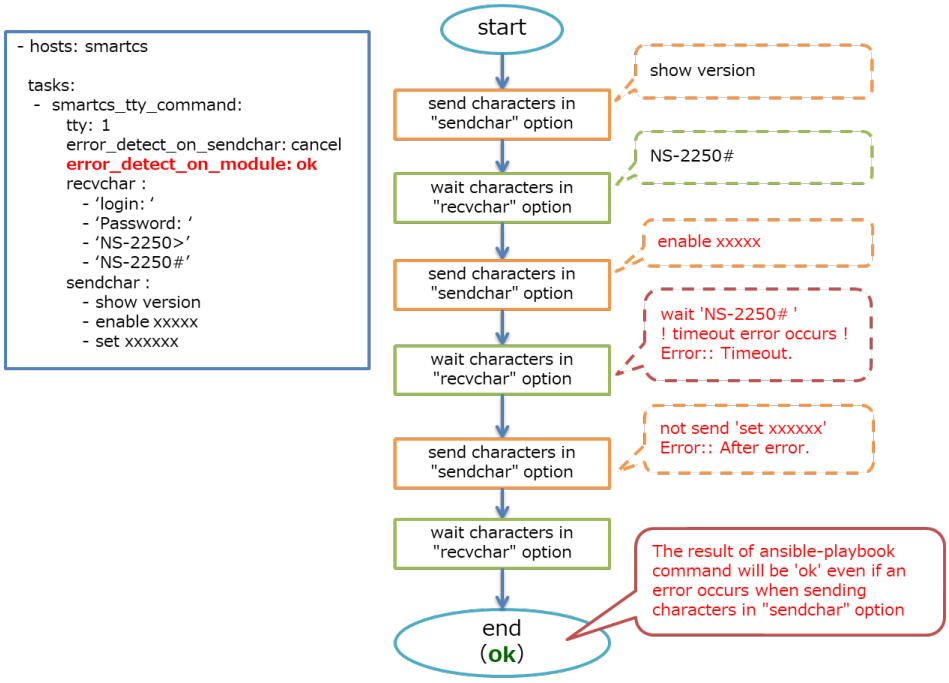
List of controllable errors	Errors occurring when sending characters configured with the sendchar option in the smartcs_tty_command module ・Error:: Timeout. ・Error:: Not allowed. ・Error:: Session limit over. ・Error:: Connection closed. ・Error:: Matched “xxx” ・Error:: After error.
Examples of uncontrollable errors	・smartcs_tty_command module option specification is incorrect. ・The CLI executed on the managed node SmartCS results in an error for some reason when executing the smartcs_tty_command module. ・An error occurs due to the SSH session conducted via Ansible (Ex.) The command timeout period configured in ansible_command_timeout has elapsed, etc.

The following table shows various combinations of the error_detect_on_module option settings, whether there is an error after sending the characters specified in sendchar/src, and ansible command

execution result.

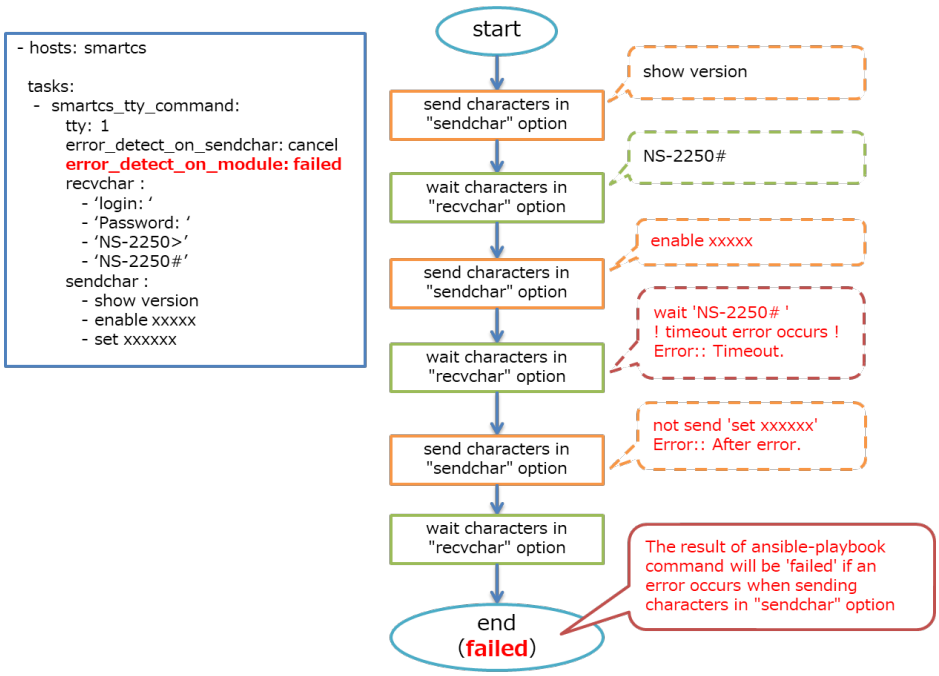
error_detect_on_module setting	Error after sending the characters	Ansible command execution result
ok	Error occurred	ok
	Error did not occur	ok
failed	Error occurred	failed
	Error did not occur	ok

1. Operation with the error_detect_on_module:ok setting



*The default value is error_detect_on_module:ok.

2. Operation with the error_detect_on_module:failed setting



(9) Output a customized return value

With the smartcs_tty_command, you can output stdout_lines_custom which differentiates each transmitted character string by setting the custom_response option.

・stdout_lines_custom

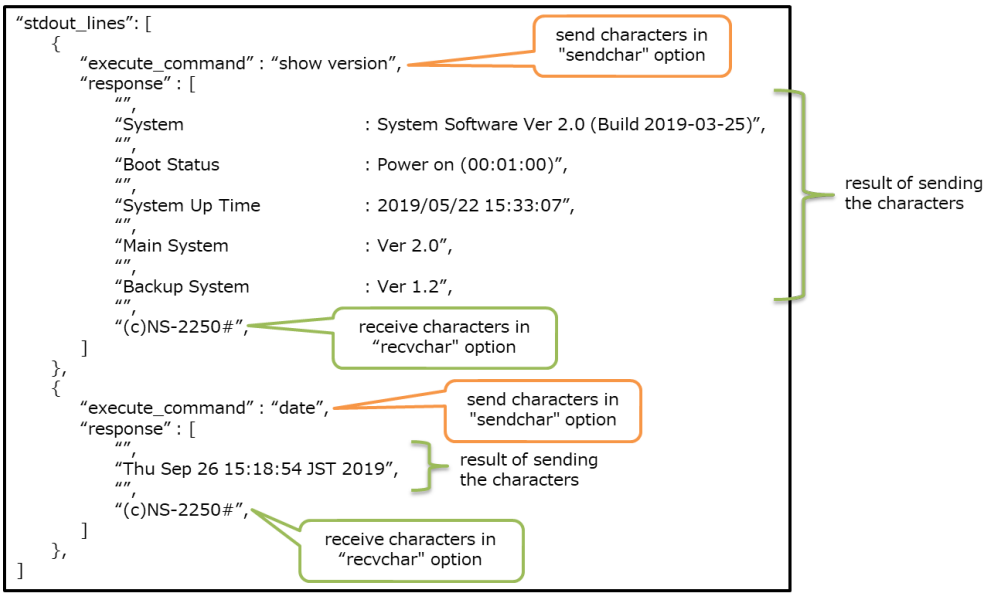
Prepare execute_command, response keys for each transmitted character string and store the following values in each key.

execute_command: transmitted character string

response : output content after sending the sendchar and before receiving the recvchar

Playbook with custom_response enables and the execution result (stdout_lines_custom)

```
- hosts: smartcs
  tasks:
   -  smartcs_tty_command:
        tty: 1
        custom_response: on
        recvchar_regex:
          - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#)'
        sendchar :
          - show version
          - date
```



custom_response option setting

・custom_response_delete_nl

Delete only the line feeds from the stdout_lines_custom response content.

```
- hosts: smartcs
  tasks:
   -  smartcs_tty_command:
        tty: 1
        custom_response: on
        custom_response_delete_nl: on
        recvchar_regex:
          - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#) '
        sendchar :
          - show version
          - date
```

```
 "stdout_lines": [
      {
         "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#"
          ],
        }
]
```

・custom_response_delete_lastline

Delete the last line from the stdout_lines_custom response content.

```
- hosts: smartcs
  tasks:
   -  smartcs_tty_command:
        tty: 1
        custom_response: on
        custom_response_delete_nl: on
        custom_response_delete_lastline: on
        recvchar_regex:
          - '(^|¥r|¥n|!)[a-zA-Z0-9_().-]*(>|#) '
        sendchar :
          - show version
          - date
```

```
 "stdout_lines": [
      {
         "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",
          ],
        }
]
```

<Supplemental>

・Depending on the timing of the console sending and receiving, there is no guarantee that input/output will be stored according to the expected format even when this function is enabled.

Since the source data which generates stdout_lines_custom is stdout and stdout_lines, specify this option after adjusting the timeout period of the transmitted character string specified in the Playbook so that the Playbook operates as intended.

8.1.6 Instructions and Directions for Use

This section provides instructions and directions for using the smartcs_tty_command module.

The smartcs_tty_command module sends and receives the specified characters to the console of the network devices connected to the SmartCS serial port. Pay attention to the following precautions during use.

(1) Module policies

1. Initial console status

   The smartcs_tty_command module does not manage or control the status of the network devices console. Depending on the last executed command, the status of the network devices console may be one of the following:
   ・Login prompt status
   ・General user group shell status
   ・Privileged user shell status
   ・Shell status for setting entry
   Create the Playbook by considering the status of the network devices console.
   Ex: always return the console to the login prompt status. Use the pre-check function, etc.

2. Console input/output results

   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 network devices.
   If you wish to control the execution result (ok/failed) of the ansible command according to the result of the CLI command executed on the console, then use the following options.
   error_recvchar_regex option
   error_detect_on_module option

(2) Precautions when creating a Playbook

1. Extend the command timeout period (ansible_command_timeout)

When executing modules such as xxx_command or xxx_config via Ansible with respect to a typical network device, it connects internally via SSH and executes each command.
Since the smartcs_tty_command module processes the execution of each command by serial communications via SmartCS, the timeout period (Playbook task time) when executing commands on the managed node via Ansible becomes longer.

Therefore, adjust the ansible_command_timeout option value according to the transmitted character string specified in sendchar/src and the timeout period within the Playbook.

*The default values for ansible_command_timeout are as follows.
Ansible2.7 system: 10 seconds
Ansible2.8 system: 30 seconds

Playbook example

```
---
- name: "get version and write"
  hosts: smartcs
  gather_facts: no

  tasks :
  - name : Configure  NS-2250  ipaddress  by Console
    seiko.smartcs.smartcs_tty_command :
      tty: 1-16
      recvchar :
      - '> '
      - '# '
      sendchar :
      - 'show version'
      - 'show ip'
      - 'write'
      - 'y'

  vars:
  - ansible_connection: ansible.netcommon.network_cli
  - ansible_network_os: seiko.smartcs.smartcs
  - ansible_user: smartcs-ansible
  - ansible_password: password
  - ansible_command_timeout: 600
```

8.2 seiko.smartcs.smartcs_command

8.2.1 Overview

Execute a status display command or maintenance command on a SmartCS device and obtain the execution result.

This module does not support the execution of setting commands. Use a smartcs_config module when configuring SmartCS.

8.2.2 Options

The following table lists the options for this module.

Option name	Required	Default	Setting range	Description
commands	○			Configure the list of commands to execute on SmartCS.
interval		1		Configure the interval time (seconds) to retry the command specified in the commands option when the conditions set in the wait_for option are not satisfied.
match		all	any all	Configure the comparison method for the conditions set in the wait_for option. When this option is set to "all," it retries the task execution when all of the conditions set in the wait_for option are satisfied. When this option is set to "any," it retries the task execution when any of the conditions are satisfied.
retries		10		Configure the number of times to retry the command execution specified in the commands option when the conditions set in the wait_for option are not satisfied.
wait_for				Configure the list of conditions for the command execution result to satisfy. The task execution fails when the conditions are not satisfied even after exceeding the number of retries set in the retries option.

8.2.3 Playbook Example

The following shows a Playbook example for this module.

1. Execute the show version command in SmartCS

```
- name: run show version on SmartCS
  seiko.smartcs.smartcs_command :
      commands: show version
```

2. Execute the show version command in SmartCS
   and check if "Ver 2.0" is included in the characters

```
- name: run show version and check to see if output contains 'Ver 2.0'
  seiko.smartcs.smartcs_command :
      commands: show version
      wait_for: result[0] contains 'Ver 2.0'
```

3. Execute multiple commands with show version and show tty in SmartCS

```
- name: run multiple command on SmartCS
  seiko.smartcs.smartcs_command :
      commands:
      - show version
      - show tty
```

4. Execute multiple commands with show version and show tty in SmartCS and check if the characters "Ver 2.0" and "1 9600" are included in each result

```
- name: run multiple commands and avaluate the output
  seiko.smartcs.smartcs_command :
      commands:
      - show version
      - show tty
      wait_for:
      - result[0] contains 'Ver 2.0'
      - result[1] contains '1 9600'
```

5. Executing the copy startup command in SmartCS
(interactive command support)

```
- name: run copy startup command on SmartCS
  seiko.smartcs.smartcs_command :
        commands:
        - command: 'copy startup 2 to startup 4'
          prompt: 'Do you really want to copy external startup2 to external startup4 \[y/n\] ?'
          answer: 'y'
```

8.2.4 Return Values

The following table lists the return values for this module.

Name	Description	Trigger	Type
stdout_lines	Output a list of command execution results separated by each line feed character.	When the command execution is successful	List
stdout	Command execution result	When the command execution is successful	List
failed_conditions	List of conditions not satisfied during the command execution	When the conditions are not established	List

8.3 seiko.smartcs.smartcs_config

8.3.1 Overview

Execute a setting command in SmartCS.

8.3.2 Options

The following table lists the options for this module.

Option name	Required	Default	Setting range	Description
backup		False	boolean value	Set the option to retrieve the backup of the current running configuration. If set to "true" (yes, y, true, etc.), the backup file is saved in the backup directory of the Playbook's save directory. If the backup directory does not exist, it creates the directory. If set to "false" (no, y, false, etc.), the backup is not retrieved.
lines				Configure the list of setting commands to execute on SmartCS. The target commands are those displayed with the show config running option.
match		line	line none	Configure the comparison method when comparing the configurations set with the lines option with respect to the configurations set in SmartCS. When this option is set to "line," the configurations set with the lines option are compared for each command with respect to the configurations set in SmartCS, and the setting command is executed if it is not configured on the device. If set to "none," the setting command is executed without comparing the configurations set in SmartCS with the configuration commands set in the lines option.

Option name	Required	Default	Setting range	Description
save_when		nerver	always never modified changed	Set the configuration save method. If this option is set to "always," the write command is executed at all times to save the configuration. If set to "modified," the show config running and show config startup execution results are compared. If there is a difference between the results, the write command is executed to save the configuration. If set to "changed," the config line specified in "lines" is not set in the execution result for show config running. If the setting is successfully configured, the write command is executed, and the configuration is saved. If set to "never," the configuration is not saved.
src				Configure the path of the file which describes the setting target configuration. This option configures an absolute file path or a relative path from the Playbook save directory. This option runs exclusively from the lines option.

8.3.3 Playbook Example

The following shows a Playbook example for this module.

1. Set the label name and baud rate of TTY1 connected to SmartCS

```
- name: configuration tty 1 settings
  seiko.smartcs.smartcs_config :
    lines:
    - set pord tty 1 label SWITCH_1
    - set tty 1 baud 38400
```

2. After setting the label name and baud rate of TTY20 connected to SmartCS,
   execute the write command if there is a difference with the config during startup.

```
- name: configuration tty 20 settings and write
  seiko.smartcs.smartcs_config :
    lines:
    - set pord tty 20 label ROUTER
    - set tty 20 baud 19200
    save_when: modified
```

3. After retrieving the backup of the current running configuration,
   configure the SmartCS hostname. Execute the write command after the configuration.

```
- name: configuration host name and get backup file
  seiko.smartcs.smartcs_config :
    lines:
    - set hostname SmartCS_TEST1
    save_when: always
    backup: yes
```

4. Issue the CLI command described in the local configuration file (config_file.txt) to SmartCS and execute the write command after configuration.

```
※config_file.txt content
==================================
  1 #
  2 set hostname NS-2250-48_2
  3
==================================

※Playbook statement example

  - name: configuration by local file and write
      seiko.smartcs.smartcs_config :
        src: config_file.txt
        save_when: changed
```

8.3.4 Return Values

The following table lists the return values for this module.

Name	Description	Trigger	Type
command	List of set configurations (same value as the updates return value)	Always	List
updates	List of set configurations (same value as the command return value)	Always	List
backup_path	Backup file absolute path	When the backup option is set to "yes"	Characters

8.4 seiko.smartcs.smartcs_facts

8.4.1 Overview

Gather the device information from SmartCS.

8.4.2 Options

The following table lists the options for this module.

Option name	Required	Default	Setting range	Description
gather_subset		!config	all default tty config	Configure the classification of device information to gather. The setting range for this option is all, default, tty, config . The following options are required, because it transitions to device administrator mode when set to "all" or "config." vars: ansible_become: yes ansible_become_method: enable ansible_become_password: xxxxx

8.4.3 Playbook Example

The following shows a Playbook example for this module.

1. Retrieve all of the information which is retrievable with the smartcs_facts module.

```
- name: Collect all facts from the SmartCS
  seiko.smartcs.smartcs_facts :
    gather_subset: all
```

2. Retrieve the SmartCS running configuration and default information.

```
- name: Collect only the config and default facts
  seiko.smartcs.smartcs_facts:
    gather_subset: config
```

3. Retrieve the SmartCS TTY information and default information.

```
- name: Collect only the tty and default facts
  seiko.smartcs.smartcs_facts:
    gather_subset: tty
```

4. Retrieve information other than the SmartCS TTY information.

```
- name: Do not collect tty facts
  seiko.smartcs.smartcs_facts:
    gather_subset: "!tty"
```

5. Retrieve the default information for the smartcs_facts module. (except config)

```
- name: Collect default facts
  seiko.smartcs.smartcs_facts:
```

8.4.4 Return Values

このモジュールの戻り値について下記に記載します。

Name	Description	Trigger	Type
ansible_net_gather_subset	List of device information classifications gathered from SmartCS	Always	Characters
ansible_net_model	SmartCS model information (model name displayed with the show version command)	Always	Characters
ansible_net_hostname	SmartCS hostname information (hostname displayed with the show ip command)	Always	Characters
ansible_net_version	SmartCS version information (version number displayed with the show version command)	Always	Characters
ansible_net_config	SmartCS running configuration	When config is enabled in the gather_subset option	Characters
ansible_net_tty	SmartCS tty setting information (baud, bitchar, flow, parity, stop, tty information displayed with the show tty command and label information displayed with the show portd command)	When tty is enabled in the gather_subset option	List

ansible_net_serialnum	SmartCS model information (serial number displayed with the show version command)	Always	Characters

Name	Description	Trigger	Type
ansible_net_bond1	SmartCS interface setting information (show ipinterface bond1 command)	Always	Hash
ansible_net_eth1	SmartCS interface setting information (show ipinterface eth1 command)	Always	Hash
ansible_net_eth2	SmartCS interface setting information (show ipinterface eth2 command)	Always	Hash

9 Limitations

9.1 Administering the SmartCS series console with “smartcs_tty_command”

When using the “SmartCS modules for Ansible” to set the SmartCS series (NS-2250, NS-2240) as the network devices and sending and receiving characters to the console with the “smartcs_tty_command”, the ansible command will exit if a CLI error (no such command, too many parameters) occurs after sending the characters configured with sendchar/src in the Playbook,.

When using the “smartcs_tty_command” to execute setting commands and status display commands on the SmartCS series, make sure that CLI errors do not occur for the characters configured with sendchar/src.

9.2 Gathering device information with gather_facts

Starting from ansible2.9, the device information can be gathered even without using the smartcs_facts module. The details are output to the ansible_facts variable by specifying "gather_facts: yes" in the Playbook. (Same information as the "gather_subset: all" specification)

Because this makes it possible to obtain the SmartCS version information without using the smartcs_facts module, it is now easier to describe a Playbook compared to before.

*This function is not a SmartCS module function enhancement but a function provided by an Ansible improved through Ansible2.9.

Playbook example

```
---
- name: "smartcs_command with fact "
  hosts: smartcs
  gather_facts: yes

  tasks :
  - name : "run show portd tty"
    seiko.smartcs.smartcs_command :
      commands:
      - show portd tty

  - name : "smartcs version"
    debug :
      var: ansible_facts.net_version

  vars:
  - ansible_connection: ansible.netcommon.network_cli
  - ansible_network_os: seiko.smartcs.smartcs
```

When using this function in “SmartCS modules for Ansible” v1.3.0 and above, the information may be handled with either (1) or (2) below.

(1) Execute smartcs_facts to obtain the information.
Execute smartcs_facts when creating a Playbook to obtain and handle the SmartCS device information. When "gather_facts: yes" is specified, the return value when "all" is specified in the gather_subset option of smartcs_facts is stored in the ansible_facts variable.

(2) Specify the FACTS_MODULES option.
The SmartCS facts information can be gathered when specifying "gather_facts: yes" by specifying the FACTS_MODULES option.
https://docs.ansible.com/ansible/latest/reference_appendices/config.html#facts-modules

The following specification is used when specifying variables in the vars section in the Playbook.

```
vars:
  - ansible_facts_modules: seiko.smartcs.smartcs_facts
```

10 Troubleshooting

This chapter explains how to handle errors that occur when using SmartCS modules for Ansible and executing a Playbook.

Each section lists the error output and a possible troubleshooting method. The troubleshooting method will not necessarily be able to resolve the error, but hopefully the information will be helpful when troubleshooting.

10.1 “Unable to connect to port 22 on x.x.x.x”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "[Errno None] Unable to connect to port 22 on x.x.x.x"
}
```

Unable to connect to the target SmartCS. Check the IP address and hostname of the SmartCS registered on the management host as well as the network between the management host and the SmartCS.

In addition, the SSH server on the SmartCS may not be enabled. Execute the following command to enable the SSH server on the SmartCS.

```
(0)NS-2250# enable sshd
```

10.2 “timed out”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "timed out"
}
```

Unable to connect to the target SmartCS, because a timeout error occurred while trying. Check the network between the management host and the SmartCS.

In addition, the SmartCS filter function may be discarding the packets. Execute the following commands to check if the packets from the management host are configured to be properly delivered to the SmartCS and add any settings as needed.

```
(0)NS-2250# show ipfilter input
(0)NS-2250# create ipfilter input accept …
```

10.3 “Error reading SSH protocol banner”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "Error reading SSH protocol banner[Errno 104] Connection reset by peer"
}
```

The following are possible causes. Check each of the SmartCS settings.

(1) Unable to connect due to an error, because access permission is not configured on the target SmartCS. Check the access permission setting on the SmartCS and add any settings as needed.

```
(0)NS-2250# show allowhost
(0)NS-2250# create allowhost …
```

(2) The target SmartCS serial port may have already reached the maximum number of connections for RW sessions. Check the portd settings and change any settings as needed.

```
(0)NS-2250# show portd tty
(0)NS-2250# set portd tty x limit rw 2 ro 3
```

10.4 “The authenticity of host ‘x.x.x.x’ can’t be established.”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "paramiko: The authenticity of host 'x.x.x.x' can't be established.\nThe ecdsa-sha2-nistp521 key fingerprint is b'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'."
}
```

The SmartCS SSH host key is not registered on the management host PC. Create an SSH connection and register the host key, or remove the comment out of "host_key_checking = False" in ansible.cfg and confirm that the check for the SSH host key is not executed.

For more details, refer to "3 Preparations."

10.5 ”Authentication failed.”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "Invalid/incorrect username/password. Authentication failed."
}
```

An authentication error is occurring when logging into SmartCS from the

management host running Ansible. Check the user name and password of the SmartCS login user.

10.6 “Bad authentication type”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "Invalid/incorrect username/password. ('Bad authentication type', ['publickey']) (allowed_types=['publickey'])"
```

An authentication error is occurring due to the wrong user authentication method when logging into SmartCS from the management host running Ansible. Match the user authentication method of the SmartCS SSH server to the management host.

```
(0)NS-2250# set sshd auth basic
```

10.7 “Unable to automatically determine host network os.”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "Unable to automatically determine host network os. Please manually configure ansible_network_os value for this host"
}
```

"smartcs" is not configured in the network OS option to connect from the management host to SmartCS. Configure "seiko.smartcs.smartcs" in the Playbook's "ansible_network_os" setting.
For more details, refer to "4 Creating a Playbook Compatible with Ansible Collections".

10.8 “unable to elevate privilege to enable mode”

```
fatal: [x.x.x.x]: FAILED! => {
    "msg": "unable to elevate privilege to enable mode, at prompt [b'\\n(2)NS-2250> '] with error: su\r\nPassword: \r\nincorrect password\r\n(2)NS-2250> "
}
```

Failed to transit to the privileged user after logging into SmartCS. Check the password specified in the Playbook's "ansible_become_password".
For more details, refer to "4 Creating a Playbook Compatible with Ansible Collections".

10.9 “command timeout triggered, timeout value is X secs.”

```
    },
    "msg": "command timeout triggered, timeout value is 10 secs.\nSee the timeout setting options in the Network Debug and Troubleshooting Guide."
}
```

The command failed to execute since timeout occurred by some causes when login to the SmartCS or executing the specified command. Regarding the causes, refer to the following documents about timeout.

Various settings regarding the Ansible timeout of the control node (ansible.cfg)
https://docs.ansible.com/ansible/latest/reference_appendices/config.html

Each setting of the network_cli connection plugin
https://docs.ansible.com/ansible/latest/collections/ansible/netcommon/network_cli_connection.html

Timeout issues in Network Debug and Troubleshooting Guide
https://docs.ansible.com/ansible/latest/network/user_guide/network_debug_troubleshooting.html#timeout-issues

10.10 “timeout value X seconds reached while trying to send～”

```
    },
    "msg": "timeout value 10 seconds reached while trying to send command: b'ttysendwaitset tty 1 timeout 15 nl cr string \"show version\"'"
}
```

A timeout occurred during the execution of a command specified in Playbook, which resulted in a command execution error.

Refer to the troubleshooting method listed in section 10.9 "command timeout triggered, timeout value is X secs." and the options for each module listed in "Chapter 8 Modules."

In case using the smartcs_tty_command module, please also check the following section.

- "8.1.2 Options" : The cmd_timeout option value setting
- "8.1.5 Explanations" : The operation of each explained option
- "8.1.6 Instructions and Directions for Use" : (2)Precautions when creating a Playbook

10.11 "Ignoring timeout(10) for smartcs_facts"

```
TASK [Gathering Facts] ************************************
[WARNING]: Ignoring timeout(10) for smartcs_facts
ok: [xxx.xxx.xxx.xxx]
```

Starting from Ansible2.9, the network module facts gathering is carried out through gather_facts, and in the case of SmartCS, the smartcs_facts module operates internally.
Therefore, it operates on the timeout value of the connection plugin (network_cli) without referencing gather_timeout (DEFAULT_GATHER_TIMEOUT), which is the timeout value for facts information gathering configured in ansible.cfg, etc.

This warning covers such details and is output by specifying "gather_facts: yes" when using each module for operating SmartCS in Ansible2.9, but there is nothing wrong with the operation or the Playbook.

11 Appendix A. Building the Ansible Environment

11.1 Building the Ansible environment with venv

When building the Ansible environment, it can be built without affecting the Python running on the control node by using the Python virtualization technology venv.

The following explains the procedure to build the Ansible environment by using venv and install the SmartCS modules for Ansible in the venv environment.

<Build environment example>

・CentOS7 (python3 package added with yum)

```
$ sudo yum install python3
```

(1) Build the virtual environment with venv

Build the virtual environment with venv in the python3 environment.

```
$ python3 –m venv smartcs-ansible
$ source smartcs-ansible/bin/activate
(smartcs-ansible)$
$
```

(2) Install Ansible and the required packages

Install Ansible and the required packages in the venv environment.

Refer to the table in "Chapter 1 Introduction 1.3 Operating environment" to install the Ansible version according to the SmartCS modules for Ansible version.

In v1.3.0 and above, Ansible v2.10 and above version which supports the Ansible Collections format must be installed. 2 types of package "ansible" and "ansible-base" are provided in Ansible v2.10 and above, but SmartCS modules for Ansible will run with either package.

The following is an operation example for building "ansible-base 2.10.6."

```
(smartcs-ansible)$ pip3 install ansible-base==2.10.6
(smartcs-ansible)$ pip3 install paramiko
(smartcs-ansible)$
```

*Supplemental (1)

Depending on the execution environment, the following type of warning may be output to indicate that the pip command must be upgraded.

As stated in the warning text, upgrade the pip command to handle this message.

```
You should consider upgrading via the 'pip install --upgrade pip' command. $
$ pip install --upgrade pip
```

*Supplemental (2)

When executing the pip command in a Proxy environment,

add the Proxy option to the pip command as shown below.

```
(smartcs-ansible)$ pip3 install ansible-base==2.10.6 --proxy x.x.x.x:xxxx
(smartcs-ansible)$ pip3 install paramiko --proxy x.x.x.x:xxxx
(smartcs-ansible)$
```

(3) Checking of the installed Ansible

Execute ansible --version to check the Ansible environment that was built on venv.

```
(smartcs-ansible)$ ansible --version
```

Check that the version of the installed Ansible is displayed.

(4) Installing the SmartCS modules for Ansible

After the Ansible environment is built, install SmartCS modules for Ansible so that the Ansible module for SmartCS will become available to use.

Because the operation details may differ depending on the version to install, refer to the following table to execute the installation.

<Provided format and install procedure>

SmartCS modules for Ansible	Provided format	Install procedure
v1.0	SEIKO Solutions original package	"Chapter 12 Appendix B. v1.0 to v1.2 Operation"
v1.1 v1.1.1		
v1.2		
v1.3.0 and above	Ansible Collections format	"Chapter 2 Installation"

11.2 Preparation for ansible.cfg

When using venv to build the Ansible environment, the ansible.cfg file required to run Ansible is not automatically generated, so the file must be prepared by yourself. Ansible.cfg can be obtained from the GitHub Ansible repository. Download and use the file according to the Ansible version which was built.

(Ex: ansible 2.10.6 (ansible-base 2.10.6) repository information)
https://github.com/ansible/ansible/blob/v2.10.6/examples/ansible.cfg

The following is an operation example which obtains ansible.cfg from GitHub.

```
$ wget https://raw.githubusercontent.com/ansible/ansible/v2.10.6/examples/ansible.cfg
$
```

The setting details described in the ansible.cfg file have a priority which depends on the location in the file, so place them according to the Ansible execution environment.

https://docs.ansible.com/ansible/latest/reference_appendices/general_precedence.html#id2

12 Appendix B. v1.0 to v1.2 Operation

12.1 Overview of the v1.0 to v1.2 operation

Seiko Solutions had been providing original package when installing “SmartCS module for Ansible” v1.0 to v1.2.

<table>
<tr><th rowspan="2">SmartCS modules for Ansible</th><th colspan="2">Control node environment</th><th colspan="2">Managed node environment SmartCS software ver.</th></tr>
<tr><th>ansible</th><th>Python</th><th>NS-2250 Series</th><th>NS-2240 series</th></tr>
<tr><td>v1.0</td><td>2.7.7</td><td rowspan="2">2.7 and above
3.6 and above</td><td>v2.0 and above</td><td rowspan="3">Not supported</td></tr>
<tr><td>v1.1
v1.1.1</td><td>2.8.4</td><td>v2.1 and above</td></tr>
<tr><td>v1.2</td><td>2.9.15</td><td>3.6.8</td><td>v2.1 and above</td></tr>
</table>

*Provided as a SEIKO Solutions original package (modules, installer).

This section explains how to install, uninstall, upgrade, and other ways to use “SmartCS modules for Ansible” v1.0 to v1.2.

12.2 Pre-installation Check

Check that Ansible is installed. If Ansible is not installed, it’s possible to install it by “yum” or “pip” command, etc. when using CentOS and other operating systems.
To build the Ansible environment, refer to "Chapter 11 Appendix A. Building the Ansible Environment."

12.3 Installation

Install “SmartCS modules for Ansible” with the following procedure.
Perform the operations as the privileged user of the Ansible management host PC as needed.
When building Ansible in a venv environment, execute the following steps after transitioning to the venv environment which was built to install the modules.

*Some “SmartCS modules for Ansible” file names may differ depending on the version.
<Version 1.0>
"smartcs_modules_for_ansible.tar.gz"... does not include its version name.
<Version 1.1 and above>
"smartcs_modules_for_ansible_vXXX.tar.gz"... includes its version name.
(The version name of version 1.1 is v110.)

(1) Unzipping “SmartCS modules for Ansible”

Place the “smartcs_modules_for_ansible_vXXX.tar.gz” file included within the downloaded file in any directory.

```
$ ls
smartcs_modules_for_ansible_vXXX.tar.gz
$
$ tar zxvf smartcs_modules_for_ansible_vXXX.tar.gz
smartcs_modules_for_ansible_vXXX/
smartcs_modules_for_ansible_vXXX/readme
smartcs_modules_for_ansible_vXXX/install_smartcs_modules.sh
smartcs_modules_for_ansible_vXXX/COPYING
smartcs_modules_for_ansible_vXXX/smartcs_modules.tar.gz
$
```

(2) Installing “SmartCS modules for Ansible”

Go to the “smartcs_modules_for_ansible_vXXX” directory created by unzipping and execute “install_smartcs_modules.sh”.
“SmartCS modules for Ansible” are installed below the Python on the management host PC where Ansible is installed.
Only the prompt appears when the modules are successfully installed.

```
$ ls
smartcs_modules_for_ansible_vXXX
smartcs_modules_for_ansible_vXXX.tar.gz
$
$ cd smartcs_modules_for_ansible_vXXX/
$
$ ls
COPYING   install_smartcs_modules.sh   readme   smartcs_modules.tar.gz
$
$ sudo ./install_smartcs_modules.sh install
$
```

(3) Installation check

Check the “SmartCS modules for Ansible” version which was installed.

```
$ ./install_smartcs_modules.sh version
RUNNING version            : 1.x (rxxxx).
$
```

12.4 Upgrading

To upgrade “SmartCS modules for Ansible”, follow the steps below.
If necessary, perform the operation as a privileged user on the Ansible management host PC.

(1) Uninstall the installed ”SmartCS modules for Ansible”
Refer to "12.5 Uninstalling" for the procedure to uninstall the modules.

(2) Upgrading Ansible
Check the version of Ansible which is supported by the new“SmartCS modules for Ansible” and perform anupgrade. For the supported versions, refer to section "1.3 Operating Environment."

(3) Installing the new “SmartCS modules for Ansible”
Refer to "12.3 Installation" for the installation procedure.
After installing the new version, use the same procedure described in "12.3 (3) Installation check" and execute the "./install_smartcs_modules.sh version" command to check the version namer.

12.5 Uninstalling

Uninstall the “SmartCS modules for Ansible” with the following procedure.
If necessary, perform the operation as a privileged user on the Ansible management host PC.

(1) Check the version

Check the installed version of “SmartCS modules for Ansible”.

```
$ ./install_smartcs_modules.sh version
RUNNING version             : 1.x (rxxxx).
$
```

(2) Uninstalling “SmartCS modules for Ansible”

Go to the “smartcs_modules_for_ansible_vXXX” directory created by unzipping of the “smartcs_modules_for_ansible_vXXX.tar.gz” file which includes the confirmed version name and execute “install_smartcs_modules.sh”.

```
$ ls
smartcs_modules_for_ansible_vXXX
smartcs_modules_for_ansible_vXXX.tar.gz
$
$ cd smartcs_modules_for_ansible_vXXX/
$
$ ls
COPYING   install_smartcs_modules.sh   readme   smartcs_modules.tar.gz
$
$ sudo ./install_smartcs_modules.sh uninstall
$
```

When upgrading Ansible, first uninstall the “SmartCS modules for Ansible” and then install the modules once again after upgrading Ansible.

12.6 Command Reference (install_smartcs_modules)

The following explains the installer command (install_smartcs_module) included with “SmartCS modules for Ansible” v1.0 to v1.2.

Perform the operations as a privileged user of the control node as needed.

Function

Install and uninstall the “SmartCS Module for Ansible” and display the version information.

Format

install_smartcs_modules.sh { install [*Ansible_Root*] | uninstall [*Ansible_Root*]| version [*Ansible_Root*]| package }

Parameters

install [*Ansible_Root*]

Install the “SmartCS modules for Ansible”.

If *Ansible_Root* is not specified, the modules are installed in the "ansible python module location" displayed when the ansible --version command is executed.

If *Ansible_Root* is specified, the files are extracted to the specified path.

uninstall [*Ansible_Root*]

Uninstall the “SmartCS modules for Ansible”.

If *Ansible_Root* is not specified, the modules are uninstalled from the "ansible python module location" displayed when the ansible --version command is executed.

If *Ansible_Root* is specified, the files are deleted from the specified path.

version [*Ansible_Root*]

Display the version number of the installed “SmartCS modules for Ansible”.

If *Ansible_Root* is not specified, it displays the version number which is installed in the "ansible python module location" shown when the ansible --version command is executed.

If *Ansible_Root* is specified, it displays the version number extracted to the specified path.

package

Display the version number of the file located in the execution path of “install_smartcs_modules.sh”.

Usage example　Installing “SmartCS modules for Ansible”

```
$ sudo ./install_smartcs_modules.sh install
$
```

Displaying the version information of the installed “SmartCS modules for Ansible”.

```
$ ./install_smartcs_modules.sh version
RUNNING version              : 1.x (rxxxx).
$
```

13 Appendix C. Handling of Various Characters in Playbooks

13.1 Specifiable Character Types

This section explains the types of characters which can be configured in the various modules included in “SmartCS modules for Ansible”.

About the table descriptive content

・hex : indicates a hexadecimal number.

・character : indicates a character which can be specified.

・text : indicates a setting method without single or double quotations when configuring an option value in a Playbook.

(Ex.)

```
sendchar :
- show version
```

・single quotation : indicates a setting method with single quotations when specifying an option value in a Playbook.

(Ex.)

```
sendchar :
- 'show version'
```

・double quotation : indicates a setting method with double quotations when specifying an option value in a Playbook.

(Ex.)

```
sendchar :
- "show version"
```

The following types of characters can be configured in the options used to send and receive characters with “smartcs_tty_command”.

○ : transmittable

x : not transmittable

Other : as indicated in the table

hex	character	sendchar			recvchar recvchar_regex error_recvchar_regex		
		text	single quotation	double quotation	text	single quotation	double quotation
0x20	SPACE	×	○	○	×	○	○
0x21	!	×	○	○	×	○	○
0x22	“	×	○	**\(0x5c) assignment**	×	○	**\(0x5c) assignment**
0x23	#	×	○	○	×	○	○
0x24	$	○	○	○	○	○	○
0x25	%	×	○	○	×	○	○
0x26	&	×	○	○	×	○	○
0x27	‘	×	**‘ (0x27) assignment**	○	×	**‘ (0x27) assignment**	○
0x28	(	○	○	○	○	○	○
0x29	)	○	○	○	○	○	○
0x2a	*	×	○	○	×	○	○
0x2b	+	○	○	○	○	○	○
0x2c	,	×	○	○	×	○	○
0x2d	-	×	○	○	×	○	○
0x2e	.	○	○	○	○	○	○
0x2f	/	○	○	○	○	○	○
0x30	0	○	○	○	○	○	○
0x31	1	○	○	○	○	○	○
0x32	2	○	○	○	○	○	○
0x33	3	○	○	○	○	○	○
0x34	4	○	○	○	○	○	○
0x35	5	○	○	○	○	○	○
0x36	6	○	○	○	○	○	○

hex	character	sendchar			recvchar recvchar_regex error_recvchar_regex		
		text	single quotation	double quotation	text	single quotation	double quotation
0x37	7	○	○	○	○	○	○
0x38	8	○	○	○	○	○	○
0x39	9	○	○	○	○	○	○
0x3a	:	×	○	○	×	○	○
0x3b	;	○	○	○	○	○	○
0x3c	<	○	○	○	○	○	○
0x3d	=	×	○	○	×	○	○
0x3e	>	×	○	○	○	○	○
0x3f	?	×	○	○	×	○	○
0x40	@	×	○	○	×	○	○
0x41	A	○	○	○	○	○	○
0x42	B	○	○	○	○	○	○
0x43	C	○	○	○	○	○	○
0x44	D	○	○	○	○	○	○
0x45	E	○	○	○	○	○	○
0x46	F	○	○	○	○	○	○
0x47	G	○	○	○	○	○	○
0x48	G	○	○	○	○	○	○
0x49	I	○	○	○	○	○	○
0x4a	J	○	○	○	○	○	○
0x4b	K	○	○	○	○	○	○
0x4c	L	○	○	○	○	○	○
0x4d	M	○	○	○	○	○	○
0x4e	N	○	○	○	○	○	○
0x4f	O	○	○	○	○	○	○
0x50	P	○	○	○	○	○	○
0x51	Q	○	○	○	○	○	○
0x52	R	○	○	○	○	○	○
0x53	S	○	○	○	○	○	○
0x54	T	○	○	○	○	○	○

hex	character	sendchar			recvchar recvchar_regex error_recvchar_regex		
		text	single quotation	double quotation	text	single quotation	double quotation
0x55	U	○	○	○	○	○	○
0x56	V	○	○	○	○	○	○
0x57	W	○	○	○	○	○	○
0x58	X	○	○	○	○	○	○
0x59	Y	○	○	○	○	○	○
0x5a	Z	○	○	○	○	○	○
0x5b	[	×	○	○	×	○	○
0x5c	\	○	○	**\(0x5c) assignment**	○	○	**\(0x5c) assignment**
0x5d	]	×	○	○	×	○	○
0x5e	^	○	○	○	○	○	○
0x5f	_	○	○	○	○	○	○
0x60	`	×	○	○	×	○	○
0x61	a	○	○	○	○	○	○
0x62	b	○	○	○	○	○	○
0x63	c	○	○	○	○	○	○
0x64	d	○	○	○	○	○	○
0x65	e	○	○	○	○	○	○
0x66	f	○	○	○	○	○	○
0x67	g	○	○	○	○	○	○
0x68	h	○	○	○	○	○	○
0x69	i	○	○	○	○	○	○
0x6a	j	○	○	○	○	○	○
0x6b	k	○	○	○	○	○	○
0x6c	l	○	○	○	○	○	○
0x6d	m	○	○	○	○	○	○
0x6e	n	○	○	○	○	○	○
0x6f	o	○	○	○	○	○	○
0x70	p	○	○	○	○	○	○
0x71	q	○	○	○	○	○	○

hex	character	sendchar			recvchar recvchar_regex error_recvchar_regex		
		text	single quotation	double quotation	text	single quotation	double quotation
0x72	r	○	○	○	○	○	○
0x73	s	○	○	○	○	○	○
0x74	t	○	○	○	○	○	○
0x75	u	○	○	○	○	○	○
0x76	u	○	○	○	○	○	○
0x77	w	○	○	○	○	○	○
0x78	x	○	○	○	○	○	○
0x79	y	○	○	○	○	○	○
0x7a	z	○	○	○	○	○	○
0x7b	{	×	○	○	×	○	○
0x7c	|	×	○	○	×	○	○
0x7d	}	×	○	○	×	○	○
0x7e	~	×	○	○	×	○	○

When specifying double quotations (0x22) and backslash (0x5c) with recvchar, recvchar_regex, and error_recvchar_regex options, they can be configured using single quotations, but the following settings are required in a Playbook when using double quotations.

```
seiko.smartcs.smartcs_tty_command:
  tty: 1
  cmd_timeout : 10
  recvchar_regex :
  - “(^|\\n|\\r)SmartCS> ”    <- example of setting the initial characters to "SmartCS> "
```

Since double quotations and backslashes are frequently used characters in regular expressions, it is recommended to use single quotations when configuring regular expressions.

13.2 Sending Various Types of Characters

As shown below, in some cases you may want to configure specific characters and commands or regular expressions in the configurable options of the various modules included in “SmartCS modules for Ansible”.

・Configuring any command (command characters that you wish to send)
- “smartcs_command” commands option
- “smartcs_config” lines option
- “smartcs_tty_command” sendchar and recvchar options

・Configuring regular expressions
- “smartcs_tty_command” recvchar_regex and error_recvchar_regex options

Examples and precautions when configuring these options are described below.

(1) Including a single quotation or comma in the option setting value

Ex. 1: configuring characters which include a single quotation

```
Version '1.0 (2019/xx/yy)'
```

When writing the characters above in a Playbook, enclose the characters within double quotation marks.

```
seiko.smartcs.smartcs_tty_command:
  tty: 1
  recvchar :
  - "Version '1.0 (2019/xx/yy)'"
  sendchar :
  -  'show version'
```

Ex. 2: configuring characters which include a comma

```
set tty 1,3,16 baud 115200
```

When writing the characters above in a Playbook, enclose the characters within double quotation marks.

```
seiko.smartcs.smartcs_config:
  lines :
  - "set tty 1,3,16 baud 115200"
```

(2) Including a double quotation or backslash (\) in the option setting value

Ex. 1: configuring characters which include a double quotation

```
set portd tty 1 label "SWITCH A"
```

When writing the setting command above in a Playbook, enclose the command within single quotation marks.

```
seiko.smartcs.smartcs_config:
  lines:
  - 'set portd tty 1 label "SWITCH A"'
```

Ex. 2: configuring characters which include a backslash

```
(\r|\n|^)SmartCS
```

When writing the setting command above in a Playbook, enclose the regular expression within single quotation marks.

```
seiko.smartcs.smartcs_tty_command:
  tty: 10
  recvchar_regex :
  - '(\r|\n|^)SmartCS'
  sendchar :
  -   'show version'
```

(3) Including both single quotation and double quotation marks in option setting value

Ex. 1: configuring characters which include single quotation and double quotation marks

```
set logd tty 10 mail 1 subject "mail test 'A'"
```

When writing the setting command above in a Playbook, enclose the entire setting value within double quotation marks and escape the double quotation marks that you wish to send with backslashes.

```
seiko.smartcs.smartcs_config:
  lines:
  - "set logd tty 10 mail 1 subject \"mail test 'A' \""
```

13.3 Configuring Regular Expressions

Regular expressions can be configured within the following options when using the “smartcs_tty_command” module.

・recvchar_regex

・error_recvchar_regex

・initial_prompt

The following table lists the regular expressions which can be configured with these options.

(1) Expressions which match a single character

.	Match any single character.
[…]	(... is any character) Match any specified single character.
[^…]	(... is any character) Match any unspecified single character.
\k	(k is a non-alphanumeric character) Match as a character.
\d	Match a single numerical character from 0 to 9.
\D	Match a single character except for \d.
\s	Match either blank character.
\S	Match a single character except for \s.
\w	Match a single alphanumeric or "_" (underscore) character.
\W	Match a single character except for \w.
\r	Match CR (0x0d).
\n	Match LF (0x0a).

(2) Add the following to express repetitive matching

*	Repetitive matching of 0 instances or more.
+	Repetitive matching of 1 instance or more.
?	0 or 1 match.
{*m*}	(*m* is an integer of 0 or more) Repetitive matching of exactly *m* times.
{*m*,}	(*m* is an integer of 0 or more) Repetitive matching of *m* times or more.
{*m*,*n*}	(*m*, *n* are integers of 0 or more) Repetitive matching of *m* to *n* times.

(3) Other expressions

(re)	(re is any regular expression) Match re.
|	Match either expression separated by this symbol.
[0-9]	Match a single numerical character from 0 to 9.
[a-z]	Match a single alphabetic character from a to z.
[A-Z]	Match a single alphabetic character from A to Z.

(4) Combined expressions

(^|\n|\r)	Match the beginning of the line.

(5) Examples of writing regular expressions

Use such expressions when you wish to wait for multiple types of prompts composed of alphabetic characters (uppercase/lowercase), numbers, symbols (_ (underscore), . (dot), and - (hyphen)).

<Matching characters>

Ex: SmartCS_01>, SmartCS_01(config)#, SmartCS_01(config-if)#, SmartCS_01(config-line)#

```
recvchar_regex :
  - “(^|\\n|\\r)[a-zA-Z0-9_.-]*(\\(config)*(-if|-line)*\\)*(>|#)”
```

13.4 Execution Result Output Characters

The following specific characters are output in the converted state shown in the table in Ansible execution results (stdout, stdout_lines).

hex	character	stdout_lines	stdout
0x09	HT(\t)	\t	\t
0X0a	LF(\n)	Double line feed	\n\n
0x0c	FF(\f)	\f	\f
0x0d	CR(\r)	Single line feed	\n
0x22	“	\”	\”
0x5c	\	\\	\\

14 Appendix D. Tips for using the “SmartCS modules for Ansible”

14.1 How to Write the File Specifying the src Option

When specifying the transmitted character string using sendchar option, there are characters which must be enclosed within single or double quotation marks. When using src option to specify the transmitted character string in an external file, it must be written without quotation marks.

Ex 1: Specifying the transmitted character string using sendchar option

```
sendchar :
  - 'somebody'
  - '__NL__'
  - 'su'
  - '__NL__'
  - 'set user testusr password'
  - ' !pass'
  - ' !pass'
  - 'exit'
  - 'exit'
```

Ex 2: Using src option to specify the transmitted character string in an external file

```
somebody
__NL__
su
__NL__
set user testusr password
!pass
!pass
exit
exit
```

14.2 Sending Characters Simultaneously to Multiple Network devices

Change the forks value in /etc/ansible/ansible.cfg so that it takes into consideration the number of network devices that you wish to simultaneously send the characters to.

For example, to send the characters simultaneously to 48 network devices connected to the serial ports of SmartCS, the forks value must be set to 48 or higher where the default value is set to 5.

15 Licenses

15.1 Third-party Software Licenses

This chapter explains the third party software licenses used in this software.

The source code for the SmartCS modules for Ansible is available on GitHub.
https://github.com/ssol-smartcs/ansible-collections

Customers who wish to request a copy of the source code modified by Seiko Solutions should contact us.

GPLv3 License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to

control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided

you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has

been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However,

nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or

hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify

or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have

permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY

APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,

reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

15.2 Ansible Collections package creation

The following explains how to create a package in the Ansible Collections format of the SmartCS modules for Ansible.
The source is available on GitHub.
After downloading the source from GitHub, you can create a package in the Ansible Collections format by executing the ansible-galaxy command in an environment that can run Ansible2.10.

```
$ git clone https://github.com/ssol-smartcs/ansible-collections/seiko.smartcs
$
$ ansible-galaxy collection build seiko.smartcs
$
```

SEIKO
セイコーソリューションズ株式会社
〒261-8507 千葉県千葉市美浜区中瀬 1-8
support@seiko-sol.co.jp