SEIKO

Operation Guide

Operation Guide

SmartCS REST API Operation Guide

First Edition

Sep 6, 2022

U00147060001

SEIKO SOLUTIONS INC.

U00147060001 Sep 2022

Table of Contents

1 Introduction

1.1 About this document

This document is the Operation Guide which summarizes the information that is required when using the SmartCS REST API functions. It lists information such as the SmartCS settings, specifications of each API resource, and how to use it, so please refer to this document when using the REST API functions.

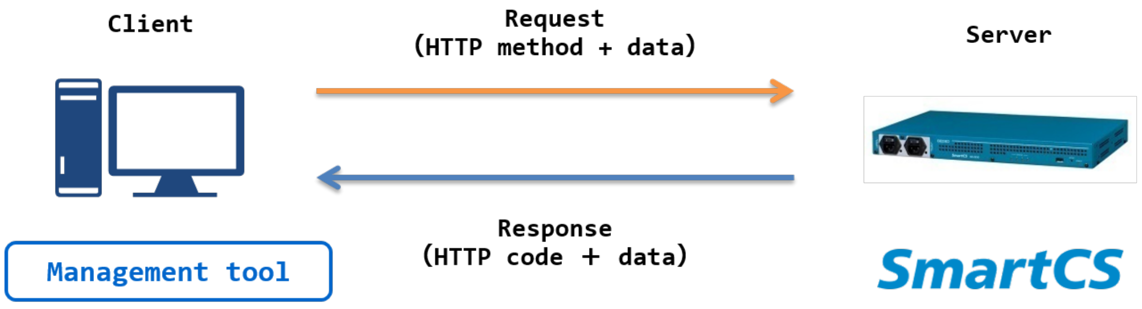
Moreover, this document covers only the SmartCS REST API functions. The "Instruction Manual" and "Command Reference" contain detailed explanations of the various SmartCS settings and CLI command details, so please refer to those documents as needed.

1.2 Function overview

This section provides an overview of the SmartCS REST API functions. SmartCS has so far executed a CLI after logging into a device via telnet or SSH for settings and information acquisition or the operation of managed nodes connected to SmartCS. By using the REST API functions, you can execute each function (processing by each API resource) from various clients and tools.

・SmartCS information and setting detail acquisition
・SmartCS settings
・Acquisition/search of TTY log information for managed nodes connected to the serial port of SmartCS
・Execution of operations on managed nodes connected to SmartCS

The REST API communicates using the HTTP protocol, requests and sends each method and transmission data from the client to specify various operations with respect to the API resources provided by SmartCS. SmartCS replies to the client with data according to the HTTP code and specified operations as a response.



<Representation of the HTTP method and CLI operation during the request>

HTTP method	Meaning	CLI example
GET	Get the API resource	show
POST	Create the API resource, etc.	create
PUT	Change or update the API resource	set, unset
DELETE	Delete the API resource	delete

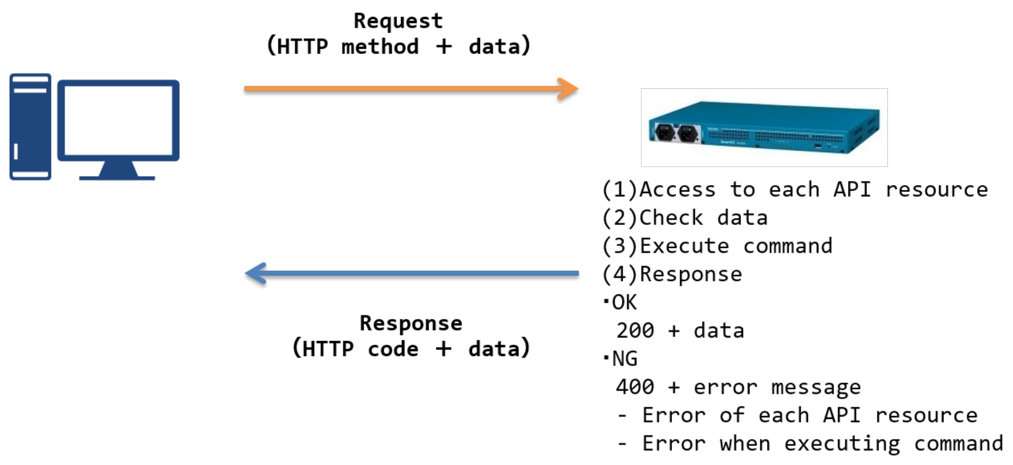
<HTTP code example during the response>

HTTP code	Meaning	Description
200	OK	Operation successful
400	Invalid request, etc.	Operation failed

1.3 SmartCS operation overview

This section provides an operation overview of the SmartCS REST API functions. The REST API functions access each API resource provided by SmartCS from the client and return a response for each function according to the request details.

A representation of the SmartCS function processes when a REST API request is received is shown in the figure and table details below.



Process	Description
Access each API resource	Accesses each API resource according to the request from the client. At that time, it also checks whether the correct method is specified and the appropriate user authority is granted, etc.
Data check	Checks whether the data is correct with respect to the specification of each API resource at the time of the request.
CLI execution	Executes the CLI command based on the request data.
Response	Returns the response according to the CLI execution result.

After the request is received, each CLI function runs and provides each function. This document explains each API resource specification without explaining the detailed specifications of each CLI. Because an error message may be output by the CLI when an operation does not work properly in some cases, refer to the "Command Reference."

1.4 Operating environment

The SmartCS supports REST API functions in version 3.0 and above. The API resource specifications for each version are as shown in the following table.

SmartCS version	Base URL
v3.0	http://<IP address>:<http port number>/api/v1/

Refer to "Chapter 4 API Resources and Methods" for each address after the base URL.

2 Preparation

2.1 SmartCS preparation

2.1.1 Enable and disable REST API functions

This section describes the SmartCS settings for using the REST API functions.

(1) Enable the HTTP/HTTPS functions

Enable the SmartCS HTTP/HTTPS functions.

Enabling the HTTP function

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

Enabling the HTTPS function

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

Setting the HTTP/HTTPS port number

```
(0)NS-2250# set http port 20080
(0)NS-2250# set https port 30443
(0)NS-2250#
```

To check whether the setting is enabled or not, check if the specified setting is enabled or the port number is changed with show service. In addition, also check whether the TCP port of the specified service is open with the show tcp command.

A maximum of eight simultaneous sessions can access HTTP/HTTPS with the REST API functions.

(2) Enable the TTY manage function

The TTY manage function must be enabled when accessing the /ttymanage API resource and performing operations on managed nodes connected to SmartCS.

Enabling the TTY manage function

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

2.1.2 User creation

This section explains the process for creating a user to access the SmartCS REST API functions. A user for accessing each API resource provided by the REST API functions is a user that belongs to the extusr group.

A user that belongs to the extusr group can only access via SSH. Such users cannot access via telnet or console.

■SmartCS user groups and executable functions

User group	Description
normal root	User group that can connect to SmartCS via telnet, ssh, or console to configure devices and acquire information.
portusr	User group that can connect to managed nodes connected to the serial port of the SmartCS via telnet or ssh and carry out operations.
extusr	User group who can be granted authority by setting. Even when being not granted authority, this group has the same authority as normal group users. The authorities which can be granted are the same command execution authority (administrator authority) as root group users and the tty manage object command execution authority (authority of tty manage function). ■normal Able to access functions within those provided by the API resources that can be executed with CLI normal user authority (show and other status display commands). ■root Able to access functions within those provided by the API resources that can be executed with CLI administrator authority (create, set, unset, delete, and other configuration commands). ■ttymanage Able to search and acquire the logs of sent and received information of managed nodes connected to the serial port and access functions that realize operations within those functions provided by the API resources.
setup verup log	User group that can connect to SmartCS via FTP or SFTP (SSH) to send and receive config files, version update files, and log files.

(1) Creating an extusr group user

Create a user that belongs to the extusr user group.

```
(0)NS-2250# create user api group extusr password
New password: xxxxx
Retype new password: xxxxx
(0)NS-2250#
```

(2) Setting the administrator authority

Set the administrator authority for the user in the created extusr group.

```
(0)NS-2250# set user api permission root on
(0)NS-2250#
```

(3) Setting the authority of tty manage function and allowing the accessible port numbers

Set the authority of tty manage function for the user in the created extusr group and allow the accessible port numbers (ex: 1-4, 16).

```
(0)NS-2250# set user api permission ttymanage on
(0)NS-2250# set user api port 1-4,16
(0)NS-2250#
```

2.1.3 Security

When using the REST API function, set and use the Firewall (ipfilter) function to strengthen security as needed.

You can restrict access by specifying the TCP port number set to HTTP/HTTPS.

2.2 Log information

2.2.1 REST API access logs

You can check the users who accessed using the REST API function with the show log webapi command.

Checking the latest 5 access logs

```
(0)NS-2250#   show log webapi 5
2022 May 31 15:32:49 [10080] login success: restapi/172.31.8.41:42266
2022 May 31 15:32:49 [10080] logout: restapi/172.31.8.41:42266
2022 May 31 15:32:49 [10080] login success: restapi/172.31.8.41:42268
2022 May 31 15:32:49 [10080] logout: restapi/172.31.8.41:42268
2022 Jun 01 23:49:03 [10080] FAILED LOGIN FROM 172.31.8.41 FOR api,
Authentication failure.
(0)NS-2250#
```

You can check which username accessed, the IP address, port number of the connection source, and whether authentication was successful or failed.

2.2.2 REST API operation logs

The SmartCS REST API functions ultimately execute the request details with the CLI. Therefore, you can check the commands executed when accessing each API resource with the show log command.

When an api user accessed /system/version API resources with the REST API functions

```
(0)NS-2250#   show log command 5
2022 Jun   1 23:55:08 somebody: show tty
2022 Jun   1 23:55:11 somebody: show user
2022 Jun   1 23:55:12 somebody: su
2022 Jun   1 23:55:20 api: show json version
2022 Jun   1 23:55:26 root: show log command 5
(0)NS-2250#
```

When an api user used the REST API functions to access /system/version API resources, you can check whether show json version was executed.

3 REST API Functions

3.1 Request

3.1.1 Base URL

The base URL which accesses the SmartCS REST API functions uses the following address.

Protocol	Base URL
HTTP	http://<IP address>:<http port number>/api/v1/
HTTPS	https://<IP address>:<https port number>/api/v1/

The IP address supports both v4 and v6.

The default value of the HTTP port number is 10080, and the default value of the HTTPS port number is 10443. Port numbers can be changed for each across a range from 1025 to 65000.

Refer to the Instruction Manual or the Command Reference for how to change the SmartCS IP address and HTTP/HTTPS port numbers.

3.1.2 HTTP method

The SmartCS REST API functions support the following HTTP methods.

Method	Description
GET	Issues a request to acquire information from the specified API resource. The operation corresponds to the show related command under the SmartCS CLI commands.
POST	Issues a request to create a new resource in the specified API. Moreover, it issues a request to execute a specific operation. The operation corresponds to the create command, etc. under the SmartCS CLI commands.
PUT	Issues a request to change/modify the specified API resource. The operation corresponds to the set/unset commands under the SmartCS CLI commands.
DELETE	Issues a request to delete the specified API resource. The operation corresponds to the create command, etc. under the SmartCS CLI commands.

The methods which can be specified for each API resource differ, and an error occurs when an unsupported method is specified. For details, refer to "Chapter 4 API Resources and Methods."

3.1.3 Parameters

Each method can specify the parameters at the time of the request.

Request	Description
GET query	Optional values can be specified as parameters for a URL query. Ex: http://<IP>:<PORT>/api/v1/users/{username} The {username} portion is optional data that can be specified as a GET query. The specifiable content differs according to each API resource.
Request body	Optional values in JSON format can be specified as parameters for the request body. At that time, specify application/json as the HTTP header content type (Content-Type).

For details, refer to "Chapter 4 API Resources and Methods."

3.2 Authentication

3.2.1 Basic authentication

The SmartCS REST API functions support Basic authentication as the form of authentication at the time of the request. Specify the username and password pair in the Authorization header. Extusr members registered in SmartCS are subject to Basic authentication.

The accessible user authority differs for each API resource. For details, refer to "Section 2.3 User Authority and API Resources."

3.3 Response

3.3.1 Status code

The SmartCS REST API functions return the following status codes as a response after accessing the API resource.

Status code	Meaning	Overview
200	OK	Status code when the REST API request was successfully processed.
400	Bad Request	Status code when the REST API request was unable to be successfully processed due to some sort of error. Check the "message" in the JSON data returned at the time of the error for the reason why the error occurred. For details about each type of error, refer to Section "3.4 Common Errors."

3.3.2 Common data

After accessing each API resource, it returns JSON formatted data as a response together with the status code. The JSON formatted data is divided into common data and response data that is unique to each API resource with the common data stored in object format with "info" as the key.

<Common data>

info	Type	Meaning
result	Numerical value	0 if the request was successfully processed. Returns a value of 1 or higher if an error occurred.
message	Character string	Stores an error message. An empty character string ("") is stored if the request was successfully processed. Error message example ・For errors concerning authentication and access authority which commonly occur in each API resource, refer to Section "3.4 Common Errors." ・For the parameters of each API resource and CLI errors which are executed through the API resources, refer to each URL error in "Chapter 4 API Resources and Methods."

Ex: if the request was successfully processed

```
{
    "info": {
        "result": 0,
        "message": ""
    },
    "systeminfo": {
        "Boot": {
            "System": {
                "Version": "3.0d",
                "Build": "2022-05-20",
```

Common data

Response data for each API resource

Ex: if the request was not successfully processed

```
{
    "info": {
        "result": 1,
        "message": "Error: Invalid request. "
    }
}
```

Common

3.4 Common error

This section explains the details of errors which commonly occur with each API resource and how to handle them when they occur.

Message	Explanation
Error: Invalid request.(400)	Displayed when there is an irregularity in the information for accessing the specified API resource.
Error: Invalid request.(404)	Displayed when the specified URL does not exist in the specified API resource.
Error: Invalid request.(405)	Displayed when the specified method is not supported by the specified API resource.
Error: Invalid request.(411)	Displayed when the required header information (content-length) is not included in the request information.
Error: Invalid request.(417)	Displayed when unsupported header information (Expect) is included in the request information.
Error: Invalid request.(500)	Displayed when an error occurs on the SmartCS web server after receiving the request.
Error: Invalid request.(501)	Displayed when an unsupported method is specified.

Message	Explanation
Error: Invalid request.(601)	Displayed when the required parameter (Authorization) is not included in the request information header.
Error: Invalid request.(602)	Displayed when the user specified in the request information header does not exist or when the setting (authority) for accessing the specified API resource is not assigned. Check the specified user information and SmartCS user settings.
Error: Invalid request.(901)	Displayed when an error occurs on the SmartCS web server after receiving the request.

When these errors occur, check the request information (URL or specified optional parameters) by referring to the specifications for each API resource.

4 API Resources and Methods

The following is a list of resources provided by the SmartCS REST API functions.

This chapter explains the specifications of each API resource (overview, requests, responses, errors, and execution examples).

Classification	URL	Method	Overview
SYSTEM	/system/version	GET	System information acquisition
USERS	/users	GET	User information (list) acquisition
		POST	User creation
	/users/{username}	GET	User information acquisition
		PUT	User information editing
		DELETE	User deletion
	/users/login	GET	Login user information acquisition
SERIAL	/serial/tty	GET	Serial information (list) acquisition
	/serial/tty/{ttylist}	GET	Serial information acquisition
		PUT	Serial information editing
	/serial/hangup/tty/{ttylist}	POST	Serial hangup
TTYMANAGE	/ttymanage	POST	Uses the TTY manage function to execute character string sending and receiving scripts on the serial port
LOG/HISTORY	/log/history/console	GET	SmartCS console log information acquisition
	/log/history/command	GET	SmartCS command log information acquisition
	/log/history/ttysend	GET	SmartCS ttysend log information acquisition
	/log/history/webapi	GET	SmartCS webapi log information acquisition
LOG/SERIAL	/log/serial/tty/{ttyno}	GET	SmartCS tty log information acquisition
	/log/serial/files/tty/{ttyno}	GET	SmartCS tty log information acquisition (DL)
	/log/serial/search/tty/{ttyno}	GET	SmartCS tty log information search

4.1 SYSTEM

4.1.1 /system/version (GET)

4.1.1.1 Overview

Acquire the system information.

4.1.1.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	This API resource is not a specifiable option.

4.1.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key name				Description
systeminfo	Boot	System	Version	Version of system software being booted
			Build	Creation date of system software being booted
			Unit	Side of system software being booted
		Status		Boot type
		Config	Unit	Save side of startup being booted
			Startup	Number of startup being booted
		ROM	Version	BootROM version
	SystemUpTime			System boot time
	HW	Model		Model name
		SerialNo		Serial number
		MAC	Local_Address	Ethernet address
			Number	Number of Ethernet addresses
		MainBoardCPU_Model		Mainboard CPU
		MainBoardCPU_Clock		Mainboard CPU frequency
		MainMemory		Memory capacity
	System	Main		Version of system software (Main side)
		Backup		Version of system software (Backup side)

4.1.1.4 Error

This resource only returns an error for a common error.

4.1.1.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/system/version

{
  "info": {
    "result": 0,
    "message": ""
  },
  "systeminfo": {
    "Boot": {
      "System": {
        "Version": "3.0",
        "Build": "2022-05-26",
        "Unit": "main"
      },
      "Status": "Reboot",
      "Config": {
        "Unit": "external",
        "Startup": "startup1"
      },
      "ROM": {
        "Version": "1.1"
      }
    },
    "SystemUpTime": "2022/05/26 15:29:22",
    "HW": {
      "Model": "NS-2250-16",
      "SerialNo": "56000050",
      "MAC": {
        "Local_Address": "XX:XX:XX:XX:XX:XX",
        "Number": "2"
      },
      "MainBoardCPU_Model": "e500v2",
      "MainBoardCPU_Clock": "533.333328MHz",
      "MainMemory": "1025264"
    },
    "System": {
      "Main": "3.0",
      "Backup": "2.2"
    }
  }
}
$
```

*The description of the execution example is formatted.

4.2 USERS

4.2.1 /users (GET)

4.2.1.1 Overview

Acquire a list of user information.

4.2.1.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	This API resource is not a specifiable option.

4.2.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key name			Description
users (Array)	name		User name
	group		Name of the group that the user belongs to
	encrypt		Encrypted password
	uid		User group ID information
	port		Serial port allow list
	permission	root	Display the administrator authority set for the extusr on: enabled off: disabled
		ttymanage	Display the authority of tty manage function set for the extusr on: enabled off: disabled
	sshkey (Array)	Element 1 [0]	Display the method
		Element 2 [1]	Display the public key

4.2.1.4 Error

This resource only returns an error for a common error.

4.2.1.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/users

{
  "info": {
    "result": 0,
    "message": ""
  },
  "users": [
    {
      "name": "root",
      "group": "root",
      "encrypt": "",
      "uid": 0,
      "port": "",
      "permission": "",
      "sshkey": ""
    },
    {
      "name": "somebody",
      "group": "normal",
      "encrypt": "",
      "uid": 100,
      "port": "",
      "permission": "",
      "sshkey": ""
    },
      "name": "api",
      "group": "extusr",
      "encrypt": "SPS.H.EC3v2a1JRKDVqU.a9k1OIcA0",
      "uid": 401,
      "port": "1-16",
      "permission": {
        "root": "on",
        "ttymanage": "on"
    }
  ]
}
$
```

*This execution example excerpts a portion of the information.

*The description of the execution example is formatted.

4.2.2 /users (POST)

4.2.2.1 Overview

Create a user.

4.2.2.2 Request

Item	Description
Accessible user authority	root
Optional parameter	Specify the JSON formatted data as the request body with the object format.

<Request body data format>

Key name	Value type	Description
name (Required)	Character string	Specify the username. ■Character length: up to 16 characters ■Character types: alphanumeric, "_", and "-" (leading character is alphabetical) ■default: none
group (required)	Character string	Specify the user group name. ■Specifiable values: normal, extusr, portusr, setup, verup, log ■default: none
password	Character string	Specify the password in plain text. ■Character length: up to 64 characters ■Character types: alphanumeric, SPACE ! # % * + , - . / : = @ _ ~ ■default: none ■Note: when both password and encrypt are set, password is prioritized.
encrypt	Character string	Specify the password with a hash value. *Value output by the SmartCS show config running command

Key name	Value type	Description
port	Character string or Numerical value	Specify the serial port numbers allowed for the port user and extusr. ■Settings ・A numerical value may be specified when specifying only one port. ・When specifying multiple ports, you can use ttylist format to specify the ports with a character string. Ex: ports 1, 2, 3, 4, 10, 16 “1-4,10,16” ・Delete the currently configured value if the value set is "". ■default: none
uid	Numerical value	Specify the user ID of the user to create. ■Settings 100 to 190: general user 401 to 410: extusr 501 to 599: port user 198: setup user 199: version upgrade user 200: port log acquisition user ■default: none (Automatically assigned when unspecified)

permission	Object	Specify the authority to grant for an extusr.
		■Settings
		Setting the administrator authority
		“root”: “on”
		Setting the authority of tty manage function
		“ttymanage”: “on”
		■default
		{
		“root”: “off”,
		“ttymanage”: “off”
		}

Key name	Value type	Description
sshkey	Array	Set the ssh public key. ■Settings ["<method>", "<public-key>"] Method is up to 20 characters Public-key is up to 720 characters ■default: none

<CLI execution order>

Execution order	CLI commands executed with a request extension
1	create user <username> group <group> [uid <userid>] [port <enable_port_list>] [{password | encrypt <string>}] *Set password when both password and encrypt are specified
2	set user <username> permission { root | ttymanage {on | off }}
3	set user <username> sshkey <method> <public-key>

4.2.2.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

4.2.2.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid json data (400)	Displayed when the request body details are not in the JSON format.
Error: Invalid request body. (400)	Displayed when optional parameters are not specified in the request body or when parameters outside of the specifications are specified.
Error: The required parameter is missing.($key) (400)	Displayed when required parameters are missing from the request body.
Error: Invalid argument value.($key) (200)	Displayed when the parameter specified in the request body cannot be correctly processed.
Error: $key contains non-usable characters. (200)	

When these errors are output, check the JSON data content of the request body.

4.2.2.5 Execution example

Request body JSON data

```
$ cat users-post.json
{
    "name": "testuser",
    "group": "extusr",
    "password": "abcdefghijklmn51",
    "encrypt": "UuSOuT.h8r6nSBV0xaeR1bRhLf9Zx/",
    "uid": 403,
    "port": "1-4",
    "permission": {
        "root": "off",
        "ttymanage": "on"
    },
    "sshkey": [
        "ssh-rsa","AAAAB3NzaC1yc2EAAAADAQABA3FO"
    ]
}
$
```

*Request body JSON data example

```
$ curl -u api:api -X POST
  -H "Content-Type: application/json"
  http://<IP>:<PORT>/api/v1/users --data @./users-post.json

{
  "info": {
    "result": 0,
    "message": ""
  }
}
$
```

*The description of the execution example is formatted.

4.2.3 /users/{username} (GET)

4.2.3.1 Overview

Acquire the specified user information.

4.2.3.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	Specify the username with a GET query.

<Parameter specification by GET query>

Parameters	Description
{username}	Specify the created username to acquire the information.

4.2.3.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key name			Description
users (Array)	name		User name
	group		Name of the group that the user belongs to
	encrypt		Encrypted password
	uid		User group ID information
	port		Serial port allow list
	permission	root	Display the administrator authority set for the extusr on: enabled off: disabled
		ttymanage	Display the authority of tty manage function set for the extusr on: enabled off: disabled
	sshkey (Array)	Element 1 [0]	Display the method
		Element 2 [1]	Display the public key

*The content is the same as the response data in Section 4.2.1.
Only the specified users are stored in the users array.

4.2.3.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid request. (400)	Displayed when an unsupported character string is specified in the username specified in the GET query. Check the username specified in the GET query.
Error: user {username} does not exist.(103) (200)	Displayed when the specified user does not exist. Check the username specified in the GET query.

4.2.3.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/users/somebody

{
  "info": {
    "result": 0,
    "message": ""
  },
  "users": [
    {
      "name": "somebody",
      "group": "normal",
      "encrypt": "",
      "uid": 100,
      "port": "",
      "permission": "",
      "sshkey": ""
    }
  ]
}

$
```

*The description of the execution example is formatted.

4.2.4/users/{username} (PUT)

4.2.4.1 Overview

Edit the specified user information.

4.2.4.2 Request

Item	Description
Accessible user authority	root
Optional parameter	Specify the username with a GET query. Specify JSON formatted data in object format as the request body of the specified user information. *The user information can be edited even only with the key name and value pair (only with the target data) where the settings are to be changed.

<Parameter specification by GET query>

Parameters	Description
{username}	Specify the created username to edit the information.

<Request body data format>

Key name	Value type	Description
password	Character string	Specify the password in plain text. ■Character length: up to 64 characters ■Character types: alphanumeric, SPACE ! # % * + , - . / : = @ _ ~ ■default: none ■Note: when both password and encrypt are set, password is prioritized.
encrypt	Character string	Specify the password with a hash value. *Value output by the SmartCS show config running command

Key name	Value type	Description
port	Character string or Numerical value	Specify the serial port numbers allowed for the port user and extusr. ■Settings ・A numerical value may be specified when specifying only one port. ・When specifying multiple ports, you can use ttylist format to specify the ports with a character string. Ex: ports 1, 2, 3, 4, 10, 16 "1-4,10,16" ・Delete the currently configured value if the value set is "".

permission	Object	Specify the authority to grant for an extusr.
		■Settings
		Setting the administrator authority
		"root": "on"
		Setting the authority of tty manage function
		"ttymanage": "on"
		■default
		{
		"root": "off",
		"ttymanage": "off"
		}

sshkey	Array	Set the ssh public key. ■Settings ["<method>", "<public-key>"] Method is up to 20 characters Public-key is up to 720 characters ・Delete the currently configured value if the value set is "". ■default: none

*An error does not occur even if name, uid, or group information is specified as the request body.

(Because the user information uses the GET query value, it is not referenced)

*Every key/value pair does not need to be specified.

Settings can be changed only with the target data.

<CLI execution order>

Execution order	CLI commands executed with a request extension
1	set user <username> {password | encrypt <string>}} *Set password when both password and encrypt are specified
2	set user <username> port <enable port_list > or unset user <username> port
3	set user <username> permission {root | ttymanage { on | off }}
4	set user <username> sshkey <method> <public-key> or unset user <username> sshkey

4.2.4.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

4.2.4.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid json data (400)	Displayed when the request body details are not in the JSON format.
Error: Invalid request. (400)	Displayed when an unsupported character string is specified in the username specified in the GET query. Check the username specified in the GET query.
Error: Invalid request body. (400)	Displayed when optional parameters are not specified in the request body or when parameters outside of the specifications are specified.
Error: Invalid argument value.($key) (200)	Displayed when the parameter specified in the request body cannot be correctly processed.
Error: $key contains non-usable characters. (200)	
Error: *Other error message (200)	Displayed when the specified user does not exist or when the parameter specified in the request body cannot be correctly processed.

When these errors are output, check the JSON data content of the request body.

4.2.4.5 Execution example

Request body JSON data

```
$ cat users-put.json
{
    "name": "testuser",
    "group": "extusr",
    "password": "abcdefghijklmn51",
    "encrypt": "UuSOuT.h8r6nSBV0xaeR1bRhLf9Zx/",
    "uid": 403,
    "port": "1-4,8,10",
    "permission": {
        "root": "off",
        "ttymanage": "on"
    },
    "sshkey": [
        "ssh-rsa","AAAAB3NzaC1yc2EAAAADAQABA3FO"
    ]
}
$
```

*Request body JSON data example

(The user information can be edited even only with the data lines where the settings are to be changed.)

```
$ curl -u api:api -X PUT -H "Content-Type: application/json"
  http://<IP>:<PORT>/api/v1/users/testuser --data @./users-put.json

{
  "info": {
    "result": 0,
    "message": ""
  }
}
$
```

*The description of the execution example is formatted.

4.2.5 /users/{username} (DELETE)

4.2.5.1 Overview

Delete the specified user.

4.2.5.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{username}	Specify the created username to delete.

4.2.5.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

4.2.5.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid request. (400)	Displayed when an unsupported character string is specified in the username specified in the GET query. Check the username specified in the GET query.
Error: user {username} does not exist.(103) (200)	Displayed when the specified user does not exist. Check the username specified in the GET query.

4.2.5.5 Execution example

```
$ curl -u api:api -X DELETE http://<IP>:<PORT>/api/v1/users/testuser1

{
  "info": {
    "result": 0,
    "message": ""
  },
}

$
```

*The description of the execution example is formatted.

4.2.6 /users/login (GET)

4.2.6.1 Overview

Acquire the information about the currently logged in user.

4.2.6.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	This API resource is not a specifiable option.

4.2.6.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key name		Description
user_login (Array)	User-Name	Logged in username
	Device	Device name or device number used to connect
	Login-Time	Login time
	Idle	Elapsed time since the last operation
	Remote-Host	Connected host IP address or name

4.2.6.4 Error

This resource only returns an error for a common error.

4.2.6.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/users/login

{
  "info": {
    "result": 0,
    "message": ""
  },
  "user_login": [
    {
      "User-Name": "somebody",
      "Device": "console",
      "Login-Time": "May 27 00:05:18",
      "Idle": "00:00",
      "Remote-Host": ""
    },
    {
      "User-Name": "api",
      "Device": "0",
      "Login-Time": "May 26 22:06:18",
      "Idle": "00:20",
      "Remote-Host": "172.31.8.41"
    }
  ]
}

$
```

*The description of the execution example is formatted.

4.3 SERIAL

4.3.1 /serial/tty (GET)

4.3.1.1 Overview

Acquire a list of the serial information for each TTY

4.3.1.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	This API resource is not a specifiable option.

4.3.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key			Description
ttylist (Array)	tty		Serial port number
	config	baud	Serial port transfer rate
		bitchar	Serial port data bit length
		parity	Serial port parity
		stop	Serial port stop bit length
		flow	Serial port flow control
		detect_dsr	DSR signal transition detection function operation setting
		label	Managed node label name
	status	DSR	Current status of the DSR signal
		CTS	Current status of the CTS signal
		DTR	Current status of the DTR signal
		RTS	Current status of the RTS signal
		CD	Current status of the CD signal
	stats	TX_Octets	Number of transmission octets
		RX_Octets	Number of reception octets
		Error_Parity	Number of reception parity errors
		Error_Framing	Number of reception framing errors
		Error_Overrun	Number of reception overruns
		Break_Count	Number of reception breaks

4.3.1.4 Error

This resource only returns an error for a common error.

4.3.1.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/serial/tty

{
  "info": {
    "result": 0,
    "message": ""
  },
  "ttylist": [
    {
      "tty": 1,
      "config": {
        "baud": 115200,
        "bitchar": 8,
        "parity": "none",
        "stop": 1,
        "flow": "none",
        "detect_dsr": "off",
        "label": "NS-2250-48"
      },
      "status": {
        "DSR": "on",
        "CTS": "on",
        "DTR": "on",
        "RTS": "on",
        "CD": "on"
      },
      "stats": {
        "TX_Octets": 0,
        "RX_Octets": 0,
        "Error_Parity": 0,
        "Error_Framing": 0,
        "Error_Overrun": 0,
        "Break_Count": 0
      }
    },
    {
      "tty": 2,
      "config": {
        "baud": 9600,
  :
  Omitted
  :
    }
  ]
}

$
```

*The description of the execution example is formatted.

4.3.2 /serial/tty/{ttylist} (GET)

4.3.2.1 Overview

Acquire the serial information for the specified TTY.

4.3.2.2 Request

Item	Description
Accessible user authority	normal root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{ttylist}	Specify the TTY number in ttylist format. Ex: specifying the tty numbers 1, 2, 3, 4, 10 1-4,10

4.3.2.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key			Description
ttylist (Array)	tty		Serial port number
	config	baud	Serial port transfer rate
		bitchar	Serial port data bit length
		parity	Serial port parity
		stop	Serial port stop bit length
		flow	Serial port flow control
		detect_dsr	DSR signal transition detection function operation setting
		label	Managed node label name
	status	DSR	Current status of the DSR signal
		CTS	Current status of the CTS signal
		DTR	Current status of the DTR signal
		RTS	Current status of the RTS signal
		CD	Current status of the CD signal
	stats	TX_Octets	Number of transmission octets
		RX_Octets	Number of reception octets
		Error_Parity	Number of reception parity errors
		Error_Framing	Number of reception framing errors
		Error_Overrun	Number of reception overruns
		Break_Count	Number of reception breaks

4.3.2.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid request. (400)	Displayed when an unsupported character string or a value that is outside of the specification is specified in the TTY list (ttylist) specified in the GET query. Check the ttylist specified in the GET query.
Error: "show json tty 1,16,17 <-- syntax error [tty number (n[-n][,n[-n]]... n=1-16 listmax=16)]" *Other CLI error messages (200)	Displayed when the specified tty number does not exist or when the parameter specified in the GET query cannot be correctly processed.

4.3.2.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/serial/tty/1,16

{
  "info": {
    "result": 0,
    "message": ""
  },
  "ttylist": [
    {
      "tty": 1,
      "config": {
        "baud": 9600,
        "bitchar": 8,
        "parity": "none",
        "stop": 1,
        "flow": "none",
        "detect_dsr": "off",
        "label": "NS-2250-48"
      },
      "status": {
        "DSR": "on",
        "CTS": "on",
        "DTR": "on",
        "RTS": "on",
        "CD": "on"
      },
      "stats": {
        "TX_Octets": 0,
        "RX_Octets": 0,
        "Error_Parity": 0,
        "Error_Framing": 0,
        "Error_Overrun": 0,
        "Break_Count": 0
      }
    },
    {
      "tty": 16,
      "config": {
        "baud": 115200,
  :
  Omitted
  :
    }
  ]
}

$
```

*The description of the execution example is formatted.

4.3.3 /serial/tty/{ttylist} (PUT)

4.3.3.1 Overview

Edit the serial information for the specified TTY.

4.3.3.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query Specify JSON formatted data in object format as the request body for each specified TTY setting. *The serial information can be edited even only with the key name and value pair (only with the target data) where the settings are to be changed.

<Parameter specification by GET query>

Parameters	Description
{ttylist}	Specify the TTY number in ttylist format. Ex: specifying the tty numbers 1, 2, 3, 4, 10 1-4,10

<Request body data format>

Key name	Value type	Description
baud	Numerical value	Specify the serial port transfer rate. ■Specifiable values: 2400, 4800, 9600, 19200, 38400, 57600, 115200 ■default: none

Key name	Value type	Description
bitchar	Numerical value	Specify the serial port data bit length. ■Specifiable values: 7,8 ■default: none
parity	Character string	Specify the serial port parity. ■Specifiable values: even, odd, none ■default: none
stop	Numerical value	Specify the serial port stop bit length. ■Specifiable values: 1, 2 ■default: none
flow	Character string	Specify the serial port flow control. ■Specifiable values: xon, rs, none ■default: none
detect_dsr	Character string	Specify the DSR signal transition detection function operation setting. ■Specifiable values: on_edge, on_polling, off ■default: none
label	Character string	Specify the managed node label name. ■Character length: 1 to 32 characters ■Character types: alphanumeric, SPACE @ _ - . ■default: none ■Notes ・Delete the currently configured value if the value set is "".

<CLI execution order>

Execution order	CLI commands executed with a request extension
1	set tty <ttylist> baud
2	set tty <ttylist> bitchar
3	set tty <ttylist> parity
4	set tty <ttylist> stop
5	set tty <ttylist> flow
6	set tty <ttylist> detect_dsr
7	set portd tty <ttylist> label <"string"> or unset portd tty < ttylist> label

4.3.3.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

4.3.3.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid json data (400)	Displayed when the request body details are not in the JSON format.
Error: Invalid request. (400)	Displayed when an unsupported character string is specified in the username specified in the GET query. Check the username specified in the GET query.
Error: Invalid argument value.($key) (200)	Displayed when the parameter specified in the request body cannot be correctly processed.
Error: $key contains non-usable characters. (200)	
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the ttylist value specified in the GET query is out of range (17 is specified for NS-2250-16).

When output, check the GET query and JSON data content of the request body.

4.3.3.5 Execution example

Request body JSON data

```
$ cat tty-put.json
{
    "baud": 9600,
    "bitchar": 8,
    "parity": "none",
    "stop": 1,
    "flow": "none",
    "detect_dsr": "off",
    "label": "SWITCH-1"
}

$
```

*Request body JSON data example

(The serial information can be edited even only with the data lines where the settings are to be changed.)

```
$ curl -u api:api -X PUT -H "Content-Type: application/json"
  http://<IP>:<PORT>/api/v1/serial/tty/16 --data @./tty-put.json

{
  "info": {
    "result": 0,
    "message": ""
  }
}

$
```

*The description of the execution example is formatted.

4.3.4 /serial/hangup/tty/{ttylist} (POST)

4.3.4.1 Overview

Backup the specified TTY.

4.3.4.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query
	POST *No request body data

<Parameter specification by GET query>

Parameters	Description
{ttylist}	Specify the TTY number in ttylist format. Ex: specifying the tty numbers 1, 2, 3, 4, 10 1-4,10

4.3.4.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

4.3.4.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid request. (400)	Displayed when an unsupported character string is specified in the username specified in the GET query. Check the username specified in the GET query.
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the ttylist value specified in the GET query is out of range (17 is specified for NS-2250-16).

4.3.4.5 Execution example

Request body JSON data

```
$ curl -u api:api -X POST -H "Content-Type: application/json" –d ""
  http://<IP>:<PORT>/api/v1/serial/hangup/tty/16

{
  "info": {
    "result": 0,
    "message": ""
  }
}

$
```

*The description of the execution example is formatted.

4.4 TTYMANAGE

4.4.1 /ttymanage (POST)

4.4.1.1 Overview

Use the TTY manage function to execute character string sending and receiving on the serial port.

4.4.1.2 Request

Item	Description
Accessible user authority	ttymanage
Optional parameter	Specify the JSON formatted data as the request body with the object format.

<Request body data format>

(1) Basic functions

Key name	Type	Description
tty (Required)	Numerical value (Character string)	Specify the tty number. ■Range: 1 to 48 ■Notes ・ttylist format is not supported ・Operates even with a character string specification of "1." ■default: none
cmd_timeout	Numerical value (Character string)	Specify the timeout value during sendchar transmission. ■Range: 1 to 30 ■Notes ・Operates even with a character string specification. ■default: 10

Key name	Type	Description
nl	Character string	Specify the line break code to assign at the end of the sendchar. ■Choices: cr, lf, crlf ■Notes: codes other than the available choices will result in an error.
recvchar	Array	Specify the character strings to wait for after the sendchar transmission. ■Number of registrations: maximum of 16 ■default: none
recvchar_regex	Array	Specify the character strings (regular expressions) to wait for after the sendchar transmission. ■Number of registrations: maximum of 8 ■default: none
sendchar	Array	Specify the character string to send. *Specify all character strings and options by enclosing them within double quotation marks. ■Number of registrations: maximum of 1024 ■Character length: 1 to 128 characters ■ Character types: alphanumeric, SPACE ! % * + , - . / : = @ _ ^ ~ ■Special transmission methods ・__NL__ Line break transmission ・__CTL__: hex Control character (one character) transmission ・ __HEX__:hexs Control character (multiple) transmission ■Transmission timing options ・__WAIT__:sec ・__NOWAIT__ ・__NOWAIT__:sec *The range of time specification options is 1 to 1800

<Request body data format>

(2) Operation upon occurrence of errors

Key name	Type	Description
error_detect_on_sendchar	Character string	When an error occurs after the sendchar transmission, specify whether to send/not send subsequent sendchar. ■Choices ・exec Transmits a sendchar even when an error occurs ・cancel Does not transmit a sendchar when an error occurs ■Notes ・Error example – TimeOut – Session limit over – Error Method – Connection closed
error_recvchar_regex	Array	If the specified character string (regular expression) is included after the sendchar transmission, specify the character string which results in an error. ■ Number of registrations: maximum of 8 ■default: none

<Request body data format>

(3) Debugging

Key name	Type	Description
ttycmd_debug	Character string	Specify the setting value regarding whether to include the debugging information in the response data. ■Choices: off, on, detail ■default: off ■Notes: codes other than the available choices will result in an error.

4.4.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
request	Request data specified in the request body
error	Number of errors that occurred
data (Array)	About the serial sending and receiving scenarios specified in the request body Transmitted data (execute_command) ･Character string Received data (response) ･Array The data stores combinations of the above in object format in an array
debug	Debug data *Only when ttycmd_debug is set to on or detail

4.4.1.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Description
Error: Invalid json data (400)	Displayed when the request body details are not in the JSON format.
Error: The required parameter is missing.($key) (400)	Displayed when required parameters are missing from the request body.
Error: The XXX option must be an array. (400)	Displayed when the parameters specified as an array type in the request body are specified in an unsupported type. (sendchar, recvchar, recvchar_regex, error_recvchar_regex)
Error: The cmd_timeout option must be an integer(range:1-30). (400)	Displayed when the setting range of the request body cmd_timeout option is incorrect.
Error: The nl option must be a string (crlf, cr, lf (default: cr)). (400)	Displayed when the setting value of the request body nl option is incorrect.
Error: XXX :sec option must be an integer(range:1-1800). (400)	Displayed when the time setting value for the transmission timing of the request body sendchar option (__WAIT__:sec, __NOWAIT__:sec) is incorrect.
Error: The configurable lines of sendchar are 1-1024. (400)	Displayed when the number of settings for the request body sendchar option is out of range.
Error: The configurable lines of recvchar are 1-16. (400)	Displayed when the number of settings for the request body recvchar option is out of range.
Error: The configurable lines of recvchar_regex are 1-8. (400)	Displayed when the number of settings for the request body recvchar_regex option is out of range.

Message (Status code)	Description
Error: The configurable lines of error_recvchar_regex are 1-8. (400)	Displayed when the number of settings for the request body error_recvchar_regex option is out of range.
Error: The error_detect_on_sendchar option must be a string (cancel, exec (default: cancel)). (400)	Displayed when the setting value for the request body error_detect_on_sendchar option is unsupported.
Error: The ttycmd_debug option must be a string (off, on, detail (default: off)). (400)	Displayed when the setting value for the request body ttycmd_debug option is unsupported.
Error: sendchar contains non-usable characters. (400)	Displayed when the setting value for the request body sendchar option includes unsupported character types.
Error: *Other error message (200)	Displayed when the setting value of the request body tty option is out of range (17 is specified for NS-2250-16) and in other cases where the specified parameter cannot be correctly processed.

When the above messages are displayed, check the JSON data content of the request body.

4.4.1.5 Execution example

Request body JSON data

```
$ cat switch_version.json
{
    "tty": 2,
    "nl": "cr",
    "cmd_timeout": 30,
    "recvchar": [
        "Switch>",
        "Switch#",
        "Press RETURN to get started."
    ],
    "recvchar_regex": [
        "[Uu]sername:",
        "[Pp]assword:",
        "(^|\\r|\\n|!)[a-zA-Z0-9_().-]*(>|#) "
    ],
    "sendchar": [
        "__NL__",
        "ssol",
        "ssol",
        "terminal length 0",
        "show version",
        "exit"
    ]
}

$
```

*This is an example of a series of scenarios ranging from logging into the Switch product connected to TTY number 2, and then executing the following commands, up to logging out from the product.

”terminal length 0”,

“show version”

*The description of the execution example is formatted.

```
$ curl -u api:api -X POST -H "Content-Type: application/json"
  http://<IP>:<PORT>/api/v1/ttymanage --data @./switch_version.json

{
  "info": {
    "result": 0,
    "message": ""
  },
  "request": {
    "tty": 2,
    "nl": "cr",
    "cmd_timeout": 30,
    "sendchar": [
      "__NL__",
      "ssol",
      "ssol",
      "terminal length 0",
      "show version",
      "exit"
    ],
    "recvchar": [
      "Switch>",
      "Switch#",
      "Press RETURN to get started."
    ],
    "recvchar_regex": [
      "[Uu]sername:",
      "[Pp]assword:",
      "(^|\\r|\\n|!)[a-zA-Z0-9_().-]*(>|#) "
    ],
    "error_recvchar_regex": [],
    "error_detect_on_sendchar": "cancel",
    "ttycmd_debug": "off"
  },
  "data": [
    {
      "execute_command": "__NL__",
      "response": [
        "Username:"
      ]
 ~~
 Omitted
 ~~
        "Press RETURN to get started.",
        ""
      ]
    }
  ],
  "error": 0
}
$
```
*A portion of the response data content (data) is omitted.
The actual serial communication operation is stored in the data portion.

4.5 LOG/HISTORY

4.5.1 /log/history/command (GET)

4.5.1.1 Overview

Acquire the SmartCS command log information.

4.5.1.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
lines	Specify the number of lines in the command log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display up to a maximum of 8192 lines in the most recent log. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.5.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
log (Array)	Display the SmartCS command log data

4.5.1.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display lines specified in the GET query. Check the content specified in the GET query.

4.5.1.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/history/command

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
    "2022 May 26 22:06:21 root: show version",
     (Omitted)
    "2022 May 26 22:06:21 root: show user"
  ]
}

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/command?lines=100

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display the 100 most recent lines

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/command?lines=all

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display all logs (maximum of 8192 lines)

4.5.2 /log/history/console (GET)

4.5.2.1 Overview

Acquire the SmartCS console log information.

4.5.2.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
lines	Specify the number of lines in the command log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display up to a maximum of 8192 lines in the most recent log. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.5.2.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
log (Array)	Display the SmartCS console log data

4.5.2.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display lines specified in the GET query. Check the content specified in the GET query.

4.5.2.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/history/console

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
    "2022 May 27 00:05:18 login: login success: somebody/console",
     (Omitted)
    "2022 May 27 00:15:18 login: logout: somebody/console"
  ]
}

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/console?lines=100

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display the 100 most recent entries

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/console?lines=all

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display all logs (maximum of 8192 lines)

4.5.3 /log/history/ttysend/tty/{ttyno} (GET)

4.5.3.1 Overview

Each command of the TTY manage function acquires the data logs sent to the TTY.

4.5.3.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{ttyno}	Specify one TTY number. ■Specification range: 1 to 48
lines	Specify the number of lines in the command log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display up to a maximum of 8192 lines in the most recent log. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.5.3.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
log (Array)	Each command of the TTY manage function displays the data logs sent to the TTY.

4.5.3.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(ttyno). (400)	Displayed when an unsupported character string or value is specified in the TTY number specified in the GET query. Check the content specified in the GET query.
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display lines specified in the GET query. Check the content specified in the GET query.
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the TTY number value specified in the GET query is out of range (17 is specified for NS-2250-16).

4.5.3.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/history/ttysend/tty/4

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
    "2022 May 12 19:01:01 restapi: enable<CR>",
     (Omitted)
    "2022 May 12 19:02:19 restapi: show running-config<CR>
  ]
}

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/ttysend/tty/4?lines=25

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display the 25 most recent lines

```
$ curl -u api:api -X GET
  http://<IP>:<PORT>/api/v1/log/history/ttysend/tty/4?lines=all

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display all logs (maximum of 8192 lines)

4.5.4 /log/history/webapi (GET)

4.5.4.1 Overview

Acquire the SmartCS webapi log information.

4.5.4.2 Request

Item	Description
Accessible user authority	root
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
lines	Specify the number of lines of the REST API log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display up to a maximum of 8192 lines in the most recent log. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.5.4.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
log (Array)	Display the SmartCS webapi log data.

4.5.4.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display entries specified in the GET query. Check the content specified in the GET query.

4.5.4.5 Execution example

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/history/webapi

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
    "2022 May 27 11:47:15 [10080] login success: api/172.31.8.41:41188",
    "2022 May 27 11:47:16 [10080] logout: api/172.31.8.41:41188"
  ]
}

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/history/webapi

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display the 100 most recent lines

```


{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display all logs (maximum of 8192 lines)

4.6 LOG/SERIAL

4.6.1 /log/serial/tty/{ttyno} (GET)

4.6.1.1 Overview

Acquire the SmartCS TTY log information.

4.6.1.2 Request

Item	Description
Accessible user authority	ttymanage
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{ttyno}	Specify one TTY number. ■Specification range: 1 to 48
lines	Specify the number of lines in the command log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display up to a maximum of 8192 lines in the most recent log. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.6.1.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
log (Array)	Display the log information for the specified TTY number saved by SmartCS.

4.6.1.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(ttyno). (400)	Displayed when an unsupported character string or value is specified in the TTY number specified in the GET query. Check the content specified in the GET query.
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display entries specified in the GET query. Check the content specified in the GET query.
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the TTY number value specified in the GET query is out of range (17 is specified for NS-2250-16).

4.6.1.5 Execution example

```
$  curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/serial/tty/2

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/serial/tty/2?lines=100

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display the 100 most recent lines

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/log/serial/tty/2?lines=all

{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
     (Omitted)
  ]
}

$
```

*Display a maximum of 8192 lines from the log file

4.6.2 /log/serial/files/tty/{ttyno} (GET)

4.6.2.1 Overview

Acquire (download) the SmartCS TTY log data.

4.6.2.2 Request

Item	Description
Accessible user authority	ttymanage
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{ttyno}	Specify one TTY number. ■Specification range: 1 to 48
lines	Specify the number of lines in the command log to display. ■Specification range ・1-1000: display the specified number of lines. Ex: if "10" is specified, it displays the 10 most recent lines. ・all: display all logs recorded to this device. ・Unspecified: display the 50 most recent lines when the parameter is unspecified.

4.6.2.3 Response

Item	Description
Format	Text data (text/plain)

<Response data>

Description
Download the log information for the specified TTY number saved by SmartCS as text data. ■File name specification (Content-Disposition attachment:filename) ・Label name_host name_ttyNN_yymmddhhmm.log – NN is the TTY number (If the TTY number is a single digit, a 0 is assigned. In the case of TTY2, the result is _tty02_) – If a SPACE character is used in the label name, it is converted to "_" (underscore character). – If the label name is not set, it becomes "host name_ttyNN_yymmddhhmm.log."

4.6.2.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(ttyno). (400)	Displayed when an unsupported character string or value is specified in the TTY number specified in the GET query. Check the content specified in the GET query.
Error: Invalid value(lines). (200)	Displayed when an unsupported character string or value is specified in the number of display entries specified in the GET query. Check the content specified in the GET query.
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the TTY number value specified in the GET query is out of range (17 is specified for NS-2250-16).

4.6.2.5 Execution example

```
$ curl -u api:api –LOJ -X GET http://<IP>:<PORT>/api/v1/log/serial/files/tty/2
$ ls
SWITCH-1_NS-2250_tty02_2205271913.log

$
```

*No options

*The description of the execution example is formatted.

```
$ curl -u api:api –LOJ -X GET
  http://<IP>:<PORT>/api/v1/log/serial/files/tty/2?lines=100
$ ls
SWITCH-1_NS-2250_tty02_2205271913.log

$
```

*Download the latest 100 lines from the log file

4.6.3 /log/serial/search/tty/{ttyno} (GET)

4.6.3.1 Overview

Search the SmartCS TTY log information.

4.6.3.2 Request

Item	Description
Accessible user authority	ttymanage
Optional parameter	GET query

<Parameter specification by GET query>

Parameters	Description
{ttyno} (Required)	Specify one TTY number. ■Specification range: 1 to 48
string (Required)	Specify the character string to search. ■Character length: 1 to 32 ■Character types: alphanumeric SPACE ! # % + , - . / : = @ _ ■Notes: when searching for a character string that includes a space, specify the space as "%20."
lines	Specify the number of targeted lines to output for the search character string. ■Specification range: 0 to 64 ·If 0 is specified, only those lines which include the search character string are output as the search result. ·If 1 is specified, the search character string + the next and previous lines (total of three lines) are output as the search result. If 64 is specified, the result is 129 lines. ■Notes: if there is no lines specification, the search character string is not output.

4.6.3.3 Response

Item	Description
Format	JSON (object format)
Common data	Refer to "3.3.2 Common data"

<Response data>

Key	Description
tty	TTY number to search for log information
label	Label name set for the TTY number to search for log information
string	Search character string
count	Number of search character strings included in the log information
data (Array)	Store the log data including the search character string contained in the log information. The search result displays up to 512 entries. An error occurs if the result exceeds 512 entries.

4.6.3.4 Error

In addition to common errors, the following errors also occur with this resource.

Message (Status code)	Explanation
Error: Invalid value(ttyno). (400)	Displayed when an unsupported character string or value is specified in the TTY number specified in the GET query.
Error: The required parameter is missing.(string) (400)	The search character string, which is a required parameter, is not specified in the GET query.
Error: The configuable length of search string is 1-32. (400)	The character length of the search character string in the GET query is out of range.
Error: string contains non-usable characters. (400)	Unsupported character types are included in the search character string of the GET query.
Error: Invalid value(lines: 0-64). (400)	The number of search character strings specified in the GET query is a value that is out of range.
Error: *Other error message (200)	Displayed when the specified parameter cannot be correctly processed such as the TTY number value specified in the GET query is out of range (17 is specified for NS-2250-16).

Check the content specified in the GET query when these messages are displayed.

4.6.3.5 Execution example

```
$ curl -u api:api -X GET
   "http://<IP>:<PORT>/api/v1/log/serial/search/tty/2?string=version"

{
  "info": {
    "result": 0,
    "message": ""
  },
  "tty": "2",
  "label": "SWITCH",
  "string": "version",
  "count": "4",
  "data": []
}

$
```

*Searching for a character string that includes "version"

*The description of the execution example is formatted.

```
$ curl -u api:api -X GET
   "http://<IP>:<PORT>/api/v1/log/serial/search/tty/2?string=version&lines=1"

{
  "info": {
    "result": 0,
    "message": ""
  },
  "tty": "2",
  "label": "SWITCH",
  "string": "version",
  "count": "4",
  "data": [
    [
      "SWITCH#",
      "SWITCH#show version",
      "XXX Software Build-Date (4/16) "
    ],
    [
      "!",
      "version 1.0",
      "xxxxxxxxxxxxxxxxxx"
    ],
    [
     (Omitted)
    ]
  ]
}

$
```

*Searching one line before and after for a character string that includes "version"

5　/ttymanage explanation

5.1　Instructions and Directions for Use

This section provides instructions and directions for using the tty manage function by specifying API resources of "/ttymanage". The tty manage function enables to send the specified characters to managed nodes connected to SmartCS and receive the command result to execute various operations.

However, please pay attention to the following precautions during use.

(1) Initial console status

The tty manage function 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 JSON data by considering the status of the network devices console.
Ex: always return the console to the login prompt status, etc.

(2) Exclusion control of console connection

The following two methods are available to operate managed notes connected to the serial port of SmartCS.

1. Access the port server by port user via telnet/ssh and directly operate managed notes.

2. Send the characters defined in JSON file as the request body to "/ttymanage" API resource using the tty manage function and operate managed notes.

Exclusion control function is available to avoid an accident such as sending unintended commands since these two methods of operation are executed at the same time.

Enabling exclusion control function of serial operation

```
(0)NS-2250# set portd service exclusive on
(0)NS-2250#
```

* Exclusion control is enabled by default.

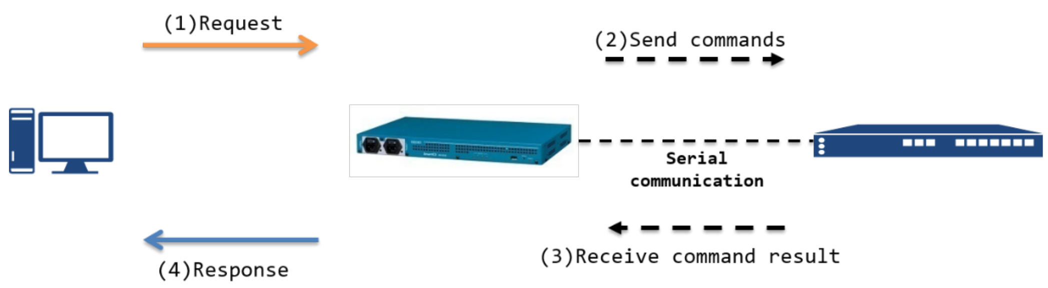
Disabling exclusion control function of serial operation

```
(0)NS-2250# set portd service exclusive off
(0)NS-2250#
```

* Use as verification to make the scenario of sending and receiving characters.

5.2 Limitations

While this content applies to the REST API functions overall, care is required regarding the timeout time of the client application particularly when the TTY manage function, which uses the /ttymanage API resource, receives a request from the client and returns a response after serial communication at its own convenience.



*It is thought that in many cases steps (2) and (3) send and receive multiple commands/execution results when processing one request.

Depending on the serial communication operation scenario, set a long timeout value on the REST API client side. Moreover, in cases where the client application timeout time cannot be changed or the setting value of the timeout time is short (when the serial communication operation scenario cannot be finished), separate the request into multiple requests.

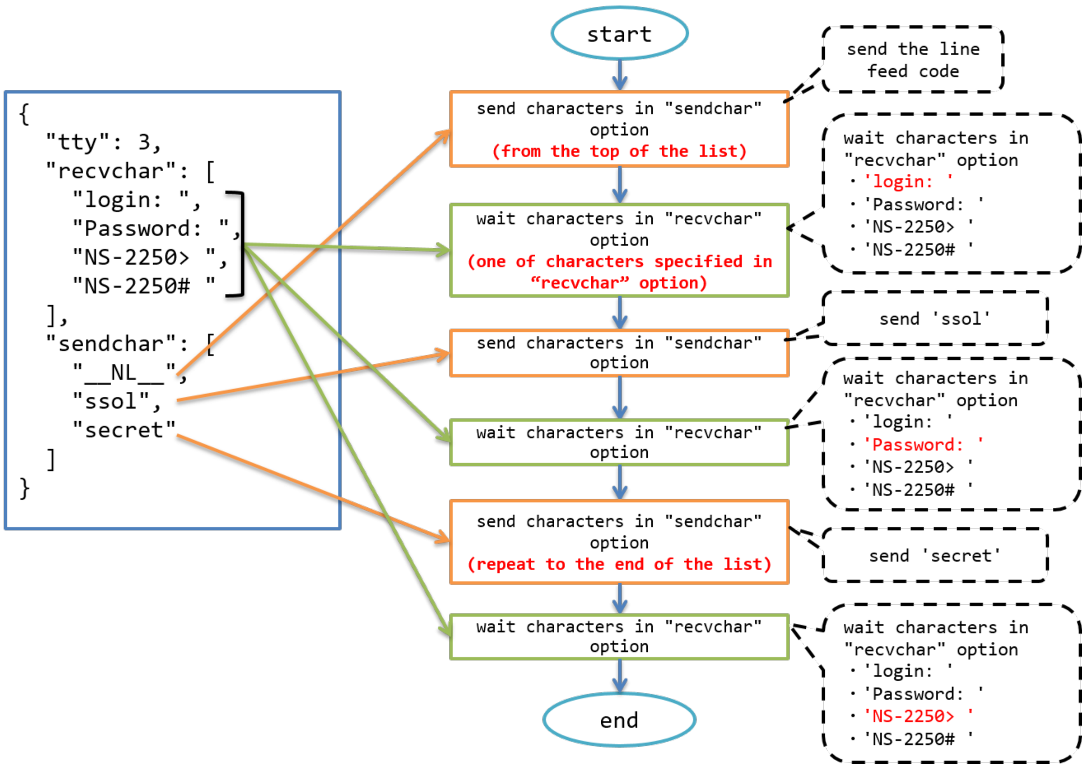
5.3 Operation under each option

This section explains the operation under each option.

5.3.1 sendchar and recvchar operation

sendchar 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 recvchar option.



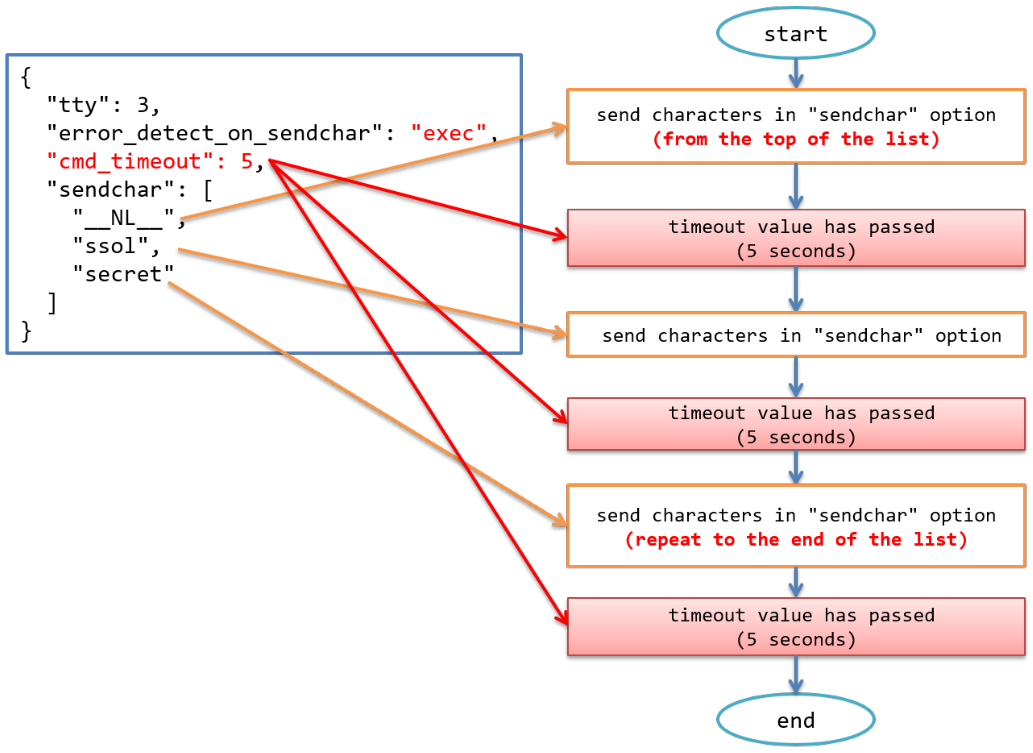
```
{
  "tty": 3,
  "recvchar": [
    "login: ",
    "Password: ",
    "NS-2250> ",
    "NS-2250# "
  ],
  "sendchar": [
    "__NL__",
    "ssol",
    "secret"
  ]
}
```

start
send characters in "sendchar" option (from the top of the list)
wait characters in "recvchar" option (one of characters specified in "recvchar" option)
send characters in "sendchar" option
wait characters in "recvchar" option
send characters in "sendchar" option (repeat to the end of the list)
wait characters in "recvchar" option
end
send the line feed code
wait characters in "recvchar" option ・'login: ' ・'Password: ' ・'NS-2250> ' ・'NS-2250# '
send 'ssol'
wait characters in "recvchar" option ・'login: ' ・'Password: ' ・'NS-2250> ' ・'NS-2250# '
send 'secret'
wait characters in "recvchar" option ・'login: ' ・'Password: ' ・'NS-2250> ' ・'NS-2250# '

5.3.2 Operation when recvchar is not configured

If recvchar (recvchar_regex) is not configured, sendchar waits until the cmd_timeout period has passed before sending the next characters.
*The only parameters which are essential for the JSON option as the request body of "/ttymaage" are the tty and sendchar options.

*The figure below shows an example of not specifying the recvchar option.



```
{
  "tty": 3,
  "error_detect_on_sendchar": "exec",
  "cmd_timeout": 5,
  "sendchar": [
    "__NL__",
    "ssol",
    "secret"
  ]
}
```

start
send characters in "sendchar" option
(from the top of the list)
timeout value has passed
(5 seconds)
send characters in "sendchar" option
timeout value has passed
(5 seconds)
send characters in "sendchar" option
(repeat to the end of the list)
timeout value has passed
(5 seconds)
end

5.3.3 Special sendchar settings

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

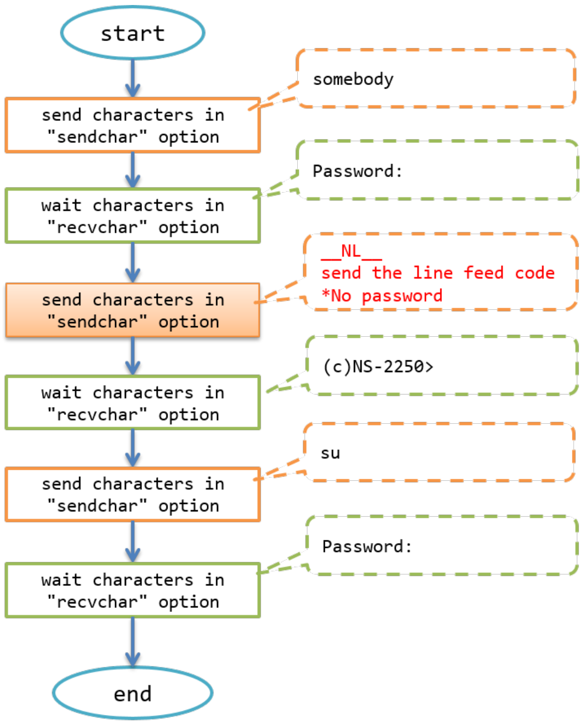
1. Send only the line feed code

   __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.

   <__NL__ option usage example>

```
{
  "tty": 1,
  "recvchar": [
    "login: ",
    "Password: ",
    "NS-2250> ",
    "NS-2250# "
  ],
  "sendchar": [
    "somebody",
    "__NL__",
    "su",
    "__NL__",
    "show version"
  ]
}
```


start
send characters in "sendchar" option
somebody
wait characters in "recvchar" option
Password:
send characters in "sendchar" option
__NL__
send the line feed code
*No password
wait characters in "recvchar" option
(c)NS-2250>
send characters in "sendchar" option
su
wait characters in "recvchar" option
Password:
end

2. Set a timeout period for each transmitted character string

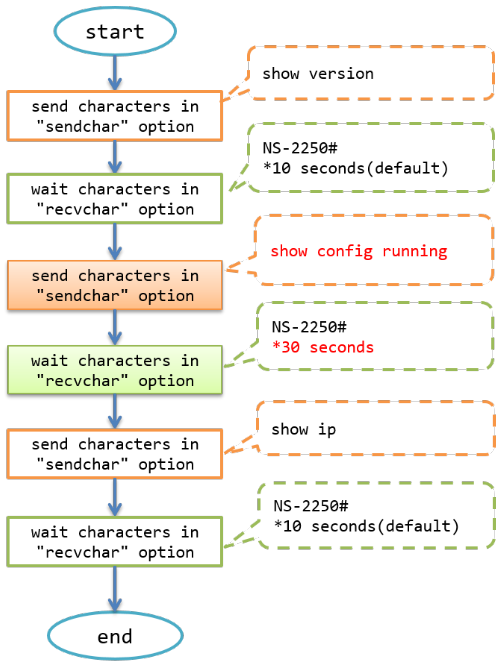
   __WAIT__:sec option

   The characters specified in sendchar 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.

   ＜__WAIT__:sec option usage example＞

```
{
  "tty": 1,
  "recvchar": [
    "login: ",
    "Password: ",
    "NS-2250> ",
    "NS-2250# "
  ],
  "sendchar": [
    "show version",
    "show config running __WAIT__:30",
    "show ip"
  ]
}
```


start
send characters in "sendchar" option
show version
wait characters in "recvchar" option
NS-2250#
*10 seconds(default)
send characters in "sendchar" option
show config running
wait characters in "recvchar" option
NS-2250#
*30 seconds
send characters in "sendchar" option
show ip
wait characters in "recvchar" option
NS-2250#
*10 seconds(default)
end

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

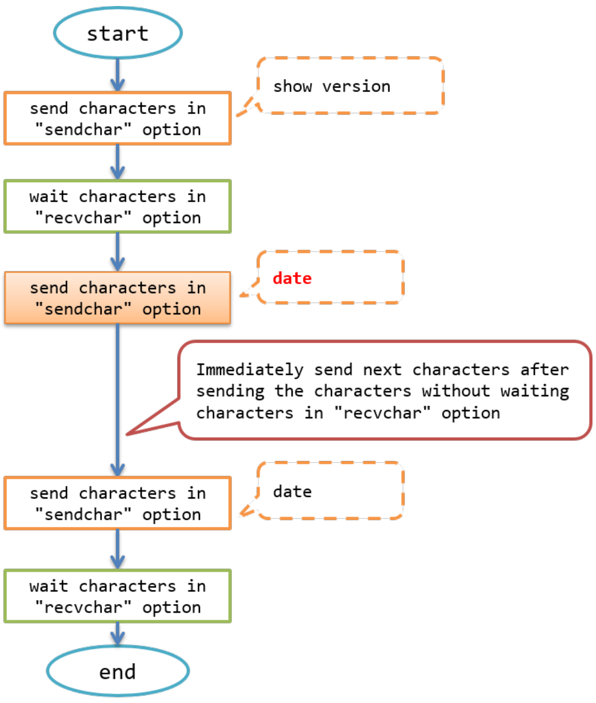
__NOWAIT__ option

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.

＜__NOWAIT__ option usage example＞

```
{
  "tty": 1,
  "cmd_timeout": 5,
  "recvchar": [
    "login: ",
    "Password: ",
    ">",
    "# "
  ],
  "sendchar": [
    "show version",
    "date __NOWAIT__",
    "date"
  ]
}
```


start
send characters in "sendchar" option
show version
wait characters in "recvchar" option
send characters in "sendchar" option
date
Immediately send next characters after sending the characters without waiting characters in "recvchar" option
send characters in "sendchar" option
date
wait characters in "recvchar" option
end

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

   __NOWAIT__：sec option

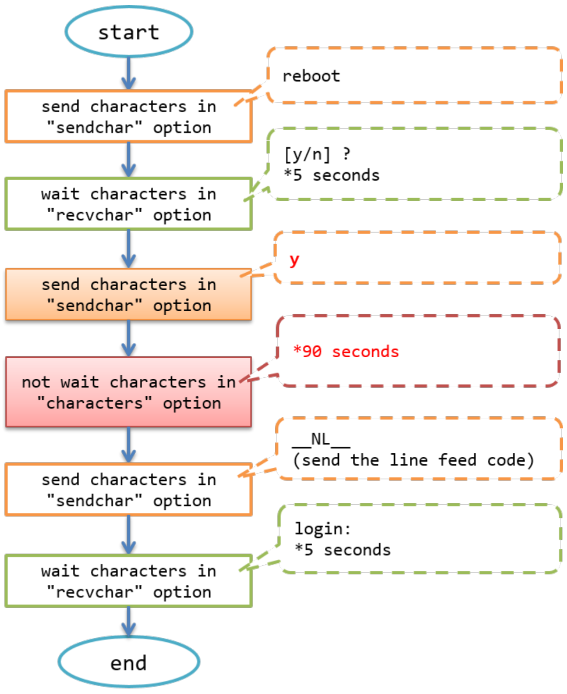
   If the recvchar setting is configured, this option checks whether or not the recvchar are included in the input/output results after sending the characters and sends the next characters if they are included.
   However, depending on the operation that you wish to perform on the network devices console, 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 as possible in the kinds of situations described above, you can set the option to not wait for recvchar for each transmitted character string.

＜__NOWAIT__:sec option usage example＞

```
{
  "tty": 1,
  "cmd_timeout": 5,
  "recvchar": [
    "login: ",
    "Password: ",
    ">",
    "#",
    "[y/n] ?"
  ],
  "sendchar": [
    "show version",
    "reboot",
    "y __NOWAIT__:90",
    "__NL__",
    "show version"
  ]
}
```



start
send characters in "sendchar" option
reboot
wait characters in "recvchar" option
[y/n] ?
*5 seconds
send characters in "sendchar" option
y
not wait characters in "characters" option
*90 seconds
send characters in "sendchar" option
__NL__
(send the line feed code)
wait characters in "recvchar" option
login:
*5 seconds
end

5. Send a control character

__CTL__ option

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

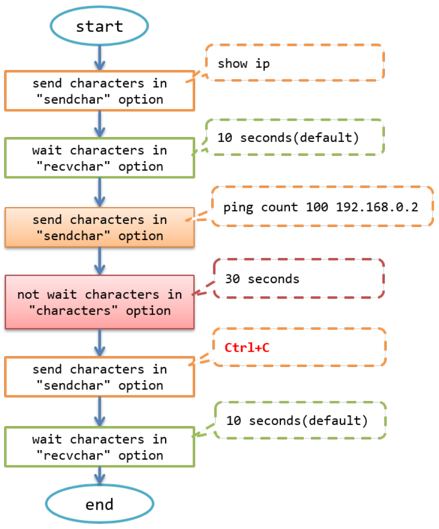
00 : [Ctrl-@] 01 : [Ctrl-A] 02 : [Ctrl-B] 03 : [Ctrl-C] 04 : [Ctrl-D] 05 : [Ctrl-E] 06 : [Ctrl-F] 07: [Ctrl-G]	08 : [Ctrl-H] 09 : [Ctrl-I] 0a : [Ctrl-J] 0b : [Ctrl-K] 0c : [Ctrl-L] 0d : [Ctrl-M] 0e : [Ctrl-N] 0f : [Ctrl-O]	10 : [Ctrl-P] 11 : [Ctrl-Q] 12 : [Ctrl-R] 13 : [Ctrl-S] 14 : [Ctrl-T] 15 : [Ctrl-U]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

*The leftmost value is the "hex" part of __CTL__:hex.

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.

＜__CTL__:hex option usage example＞

```
{
  "tty": 1,
  "cmd_timeout": 5,
  "recvchar": [
    "login: ",
    "Password: ",
    ">",
    "#"
  ],
  "sendchar": [
    "show ip",
    "ping count 100 192.168.0.2 __NOWAIT__:30",
    "__CTL__:03"
  ]
}
```

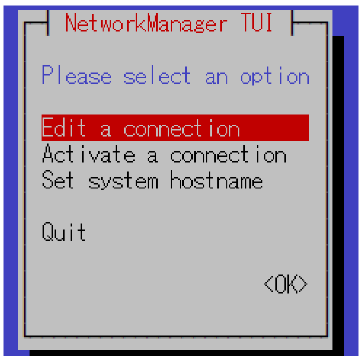

start
send characters in "sendchar" option
show ip
wait characters in "recvchar" option
10 seconds(default)
send characters in "sendchar" option
ping count 100 192.168.0.2
not wait characters in "characters" option
30 seconds
send characters in "sendchar" option
Ctrl+C
wait characters in "recvchar" option
10 seconds(default)
end

6. Sending multiple control characters together

__HEX__ option

When sending multiple control characters together, specify "__HEX__:hex" in sendchar. The range of control characters which can be sent is from 00 to 7F. The __HEX__ option does not wait for the received character string specified in recvchar.
The following types of situations are expected for sending control characters together via the console.

Cursor operations (up/down cursor movement, etc.) are performed when using the Linux nmtui command, etc. in a terminal emulator

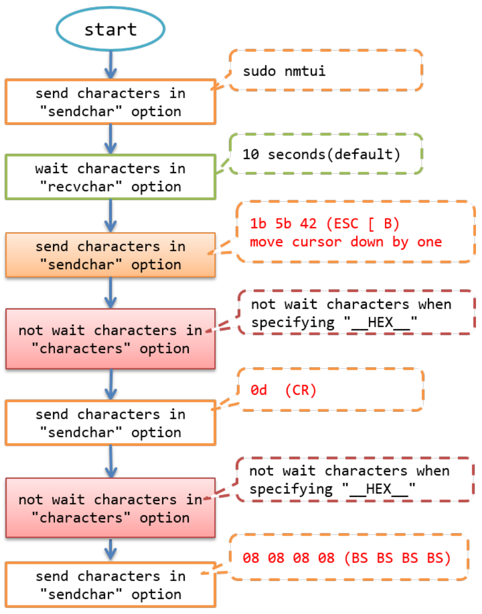


```
┤ NetworkManager TUI ├

Please select an option

Edit a connection
Activate a connection
Set system hostname

Quit

                  <OK>
```

Directly specifying and sending character codes when sending an unsupported character string with sendchar

<__HEX__:hex option usage example>

```
{
  "tty": 16,
  "recvchar": [
    "login: ",
    "Password: ",
    "~]$ ",
    "]# "
  ],
  "sendchar": [
    "sudo nmtui __NOWAIT__",
    "__HEX__: 1b 5b 42",
    "__HEX__: 0d",
    "__HEX__: 08 08 08 08",
    "__HEX__: 54 65 73 74",
    "__HEX__: 1b 5b 42",
    "__HEX__: 0d",
    "exit"
  ]
}
```



start
send characters in "sendchar" option
sudo nmtui
wait characters in "recvchar" option
10 seconds(default)
send characters in "sendchar" option
1b 5b 42 (ESC [B) move cursor down by one
not wait characters in "characters" option
not wait characters when specifying "__HEX__"
send characters in "sendchar" option
0d (CR)
not wait characters in "characters" option
not wait characters when specifying "__HEX__"
send characters in "sendchar" option
08 08 08 08 (BS BS BS BS)

5.3.4error_detect_on_sendchar operation

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

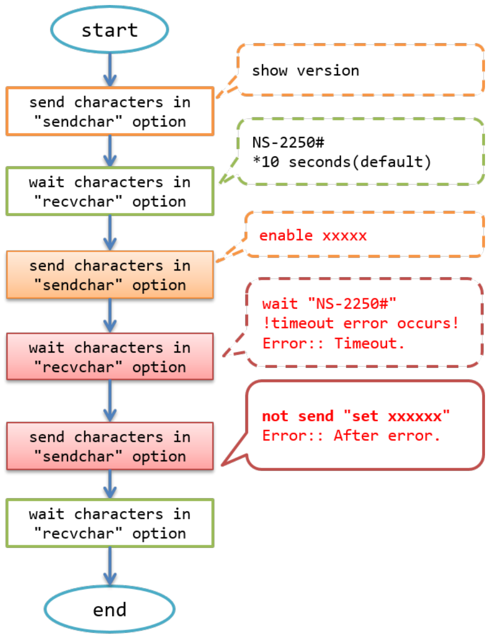
＜Causes of character transmission errors＞

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

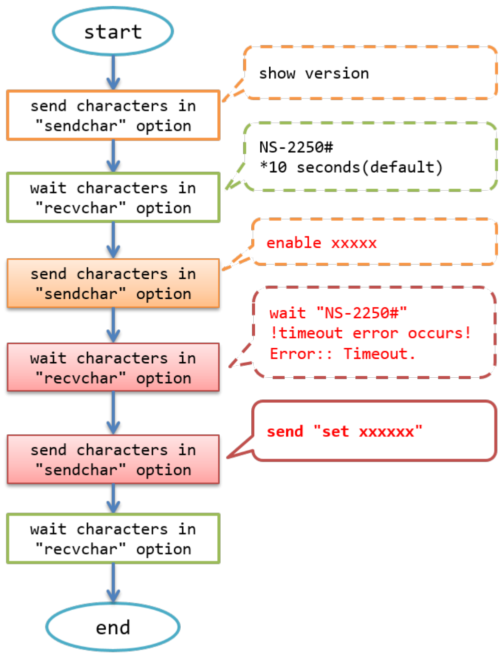
```
{
  "tty": 1,
  "error_detect_on_sendchar": "cancel",
  "recvchar": [
    "login: ",
    "Password: ",
    "NS-2250>",
    "NS-2250#"
  ],
  "sendchar": [
    "show version",
    "enable xxxxx",
    "set xxxxx"
  ]
}
```


start
send characters in "sendchar" option
show version
wait characters in "recvchar" option
NS-2250#
*10 seconds(default)
send characters in "sendchar" option
enable xxxxx
wait characters in "recvchar" option
wait "NS-2250#"
!timeout error occurs!
Error:: Timeout.
send characters in "sendchar" option
not send "set xxxxxx"
Error:: After error.
wait characters in "recvchar" option
end

*The default value is error_detect_on_sendchar:cancel.

2. Operation when error_detect_on_sendchar:exec is configured

```
{
  "tty": 1,
  "error_detect_on_sendchar": "exec",
  "recvchar": [
    "login: ",
    "Password: ",
    "NS-2250>",
    "NS-2250#"
  ],
  "sendchar": [
    "show version",
    "enable xxxxx",
    "set xxxxx"
  ]
}
```


start
send characters in "sendchar" option
show version
wait characters in "recvchar" option
NS-2250#
*10 seconds(default)
send characters in "sendchar" option
enable xxxxx
wait characters in "recvchar" option
wait "NS-2250#"
!timeout error occurs!
Error:: Timeout.
send characters in "sendchar" option
send "set xxxxxx"
wait characters in "recvchar" option
end

5.4 List of sendchar sending methods

The following table shows combinations of sendchar 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.

Configuration method	Notes
__HEX__: 54 65 73 74	Send various types of data such as control characters and control codes in hexadecimal or ASCII character strings.
__HEX__: 54 65 73 74 __WAIT__:sec	Wait for recvchar for the configured time after sending the specified data in hexadecimal.
__HEX__: 54 65 73 74 __NOWAIT__	Immediately send the next character string after sending the specified data in hexadecimal.
__HEX__: 54 65 73 74 __NOWAIT__: sec	Wait only the configured time after sending the specified data in hexadecimal.

*The "show version" portion is an example of specifying some sort of transmission character string.
*The "03" specified in "__CTL__" is an example of specifying Ctrl+C.
*The "(54 65 73 74)" specified in "__HEX__" is an example of specifying "Test" in ASCII.

*When "__HEX__" is specified, it does not wait for the character string specified in recvchar.

5.5 Configuring Regular Expressions

Regular expressions can be configured within the following options when creating the JSON data to operate console of managed nodes specifying API resource “/ttymanage”.
・recvchar_regex
・error_recvchar_regex

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)*\\)*(>|#)”
```

6 Appendix A Accessible API Resources for Each User Authority

A list of the API resources that are accessible according to each level of authority granted to extusr members is shown below.

〇: accessible

x: not accessible

API resource	Method	Authority granted to an extusr		
		normal	root	ttymanage
/system/version	GET	〇	〇	×
/users	GET	〇	〇	×
	POST	×	〇	×
/users/{username}	GET	〇	〇	×
	PUT	×	〇	×
	DELETE	×	〇	×
/users/login	GET	〇	〇	×
/serial/tty	GET	〇	〇	×
/serial/tty/{ttylist}	GET	〇	〇	×
	PUT	×	〇	×
/serial/hangup/tty/{ttylist}	POST	×	〇	×
/ttymanage	POST	×	×	〇
/log/history/command	GET	×	〇	×
/log/history/console	GET	×	〇	×
/log/history/ttysend/tty/{ttyno}	GET	×	〇	×
/log/history/webapi	GET	×	〇	×
/log/serial/tty/{ttyno}	GET	×	×	〇
/log/serial/files/tty/{ttyno}	GET	×	×	〇
/log/serial/search/tty/{ttyno}	GET	×	×	〇

SEIKO SOLUTIONS INC.

8, Nakase 1-chome, Mihama-ku, Chiba-shi, Chiba 261-8507, Japan
ns-global-support@seiko-sol.co.jp
https://www.seiko-sol.co.jp/en/