EW-WSN-FAN
Command Application
User's Manual

(Version 1.0.56.60)

Revision History

Date	Version	Revised content
2016/09/29	1.0.0	Initial version
~~ Omitted ~~~		
2020/10/15	1.0.56.58	[2.2.2 Command-line] Added command line details. [2.2.3 Data transmission command] Removed the guide to the udpst command for the udpst command argument data string. Added a command line guide for the udpst command argument data string. [3.3.3(1) leaseip] Added display format and change of display format. [3.3.3(2) leaserng] Added the setting address range of leaserng command. [3.3.2(7) macf] Added display format and change of display format. [3.3.2(10) netname] Added command line guide for input argument data (data string). Added display description of string literals for display strings. [3.3.2(14) tcpcon] Corrected the error message when connection is abnormal. [3.3.2(16) rantsw] Changed platform-dependent description. [3.3.3(3) nodef] Added display format and change of display format. [3.3.3(4) fnode] Corrected the description of command arguments. [3.3.4(5) chconfig] Changed the description of the display value for the display format content domain. [3.3.4(8) nebr] [3.3.5(2) leased] Added display format and change of display format. [3.3.6(2) udpst] Added a command line guide for the udpst command input argument data (data string). Changed the display of command usage examples. [3.3.6(3) tcps] Corrected the error message when connection is abnormal. [3.3.6(4) ping] Added the operation explanation when 0 is specified for the argument cnt. [3.3.7(1) udpopts] Added the maximum number of ports that can be registered for listen_port and listen_port_text. Changed the deletion settings for listen_port and listen_port_text. Changed and added the operation description when deleting listen_port and listen_port_text. Changed and added the operation description for send_port and send_port_text. [3.3.7(2) tcpopts] Corrected the range value of argument minutes for idle_minutes. Added the maximum number of ports that can be registered for listen_port. Change the setting value at the time of deletion for listen_port. Changed and added the operation description when deleting listen_port. [3.3.8(2) save] [3.3.8(3) clear] Changed the JSON display format.
2020/12/24	1.0.56.60	Changed the version notation. [3.3.2(17) rfec] [3.3.2(18) sleep] Added new command.

Table of contents

License and disclaimer

1. The contents of this material are subject to change without prior notice.
2. Reprinting or copying of this material requires the Company's permission.
3. Regarding the use of the information contained in this material, the Company does not guarantee, implement, or license the intellectual property rights or other rights owned by the Company or a third party.
4. The Company shall not be liable for any damages caused by the use of the information contained in this material or infringement of the rights owned by a third party.

Trademarks

1. "Wi-SUN" is a registered trademark of Wi-SUN Alliance.
2. Other company names and product names are trademarks or registered trademarks of each company.

1. Overview

The EW-WSN-FAN command application runs on EW-WSN-FAN, which is a Wi-SUN (FAN) protocol stack, and it provides a user interface with protocol stack through command-style interaction with a simple shell.

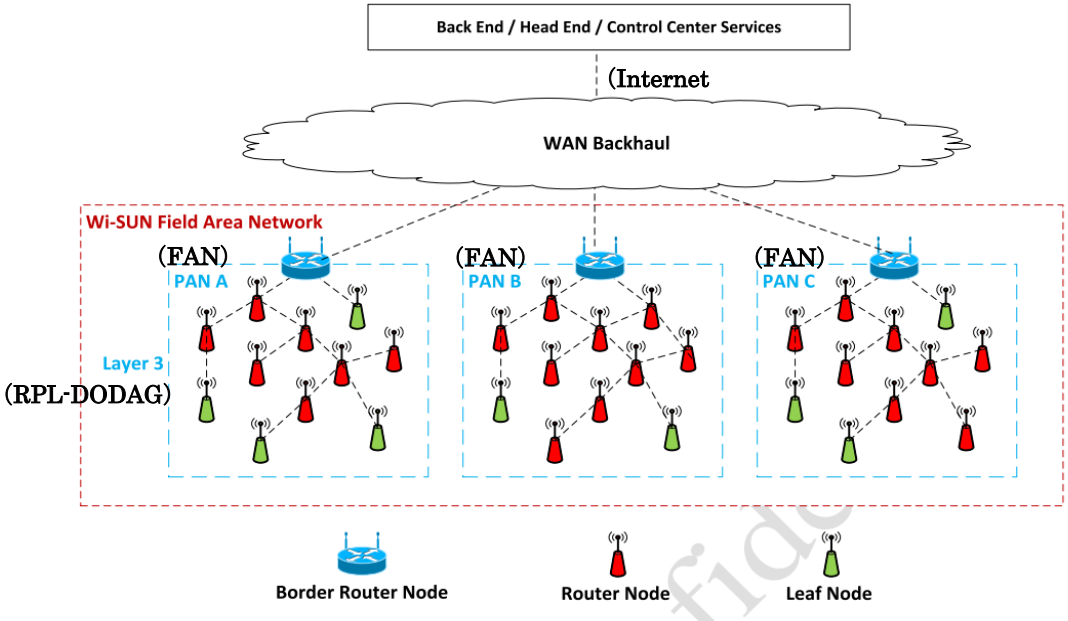
This manual describes how to use the commands of EW-WSN-FAN command application.
The command application uses the input and output from a serial port to execute various commands using ASCII commands and control the EW-WSN-FAN protocol stack.

Refer to Wi-SUN Alliance document for the details of Wi-SUN standard.

1.1 Terminology

The terms used in this manual are described as follows.

Terms	Description
Wi-SUN Module	A firmware module with installed EW-WSN-FAN.
User	A user application that operates the Wi-SUN module.
FAN	An abbreviation for Field Area Network. In IEEE802.15.4, it is expressed as PAN (Personal Area Network). However in PAN, it is expressed as Field Area Network, which means"outdoor", as opposed to HAN (Home Area Network), which means "indoor".
RPL	A Layer3 routing network protocol standardized by Wi-SUN-FAN. It is mainly the IPv6 Routing Protocol for Low-Power and Lossy Networks defined in RFC6550.
Border Router	A backbone terminal of the network tree in FAN, and a gateway terminal with WAN on the Internet. The RPL controls the source routing information of all terminals in FAN.
Router	A router terminal in FAN. In the FAN network tree, it is a terminal that performs relay communication between the master terminal, which is the Border or the upper Router, and the slave, which is the Leaf or the lower Router.
Leaf	A terminal node in FAN.
DODAG	This represents a network graph (network structure) in RPL. An abbreviation for Destination-Oriented Directed Acyclic Graph. It means a graph with only one root among non-closing directed graphs.



1.2 Restrictions

1. The functional restrictions depending on the platform are as follows.

Platform (BSP)	Channel Hopping Function (Frequency Hopping Function)	Authentication Function (EAPOL/RADIUS)	TCP Data Transmission and Reception Function
BP35C4(ROHM)	✓	✓	✓
BP35C5(ROHM)	✓	✓	✓

2. Refer to “3.3.1(7) baud” for the serial communication speed supported by the platform.
3. Refer to “3.3.2(3) chrate” for the supported PHY rates by platform.
4. The frequency setting is based on ARIB STD-T108.
5. If the number of hops increases in multi-hop communication and the number of nodes that relay packets increases, packets may be lost on the way.
6. Regarding the command execution of EW-WSN-FAN command application, the following command input is not accepted during the execution.
7. In the user data communication commands "udps" and "tcps", and "udpr" and "tcpr" messages that output received data, the size of user data that can be sent and received is 1024 Octets in binary (2048 bytes for ASCII data conversion)
8. In the user data communication command "udpst", and the "udprt" message that outputs received data, the size of user data that can be sent and received is 1024 bytes in ASCII data.
9. The RADIUS server certificate used to authenticate the operation uses a fixed certificate in this version.

2. EW-WSN Command Application

2.1 Starting the application

Writing the EW-WSN firmware to a wireless module such as USB dongle will start the command application via serial port.

2.2 Interface

2.2.1 Serial port settings

The basic serial port settings are described below.

Parameter	Description
Baud rate	115200bps(*1)
Stop bit	8bit
Parity	None
Flow control	None
New line code	Receive = LF, Send = CR

* 1 Baud rate may vary depending on the platform.
Refer to “3.3.1(7) baud” for the serial baud rate that can be set depending on the platform.

By default, the serial interface has echoback enabled.
If echoback is not required such as when the user operates the serial, it can be disabled with a command.
When echo back is enabled, one character can be deleted by "Back Space".

2.2.2 Command-line

In this manual, “command line” is described as a command character string which is formed with the following format: "command" and "command argument value".
Refer to “3 Commands” for the details on various commands using the command line.

When a command can be entered, the command prompt ">" is displayed.
Various commands are executed by terminating the ASCII character string of command line formed by the command and the argument value with a newline character.
The execution result after executing the command is outputted as ASCII character string. (Specific commands are excluded.)

A command will be executed when a <new line> is entered.
A command and command argument value are alphanumeric and symbolic character strings.

When a command processes the argument as a numerical input, the input character string is converted to numerical value and the argument is processed.

Use half-width space (' '), tab character, or comma (',') as delimiter between the command and character string of each argument value.
Use delimiter when entering character string of each argument value of a command consecutively.

Enter the command using the following format:

<Enter by separating with half-width space (' ')>

```
> <command> [argument] [argument]...[argument]<new line>
```

<Enter by separating with comma (',')>

```
> <command> [argument],[argument],...,[argument]<new line>
```

* Command input in this manual is separated by spaces.

Character string input of argument values in string literal enclosed in double quotation marks ('"') can also be entered.
(In this manual, a character string enclosed in double quotation marks ('"') is described as "string literal".)

On the command line, when entering the above delimiter (half-width space, tab character, or comma) with one argument value as a character string, enter it as string literal.

<Enter a character string including the delimiter as one argument>

```
> <command> "abc def,ghi" ... [argument]<new line>
```

* A command (command name) is not possible to enter in string literal on the command line.
* An input of delimiter that is not a string literal is the input to character string of the next argument value, assuming that the character string input of argument value is separated.

Entering control characters with escape sequence symbols with an escape character ('\') on the command line is only allowed within string literals. (The escape character ('\') not included in string literal is assumed to be the backslash character ('\\').)

Below is applied to the escape sequence of command line character string and mutual conversion to the control code.

Escape Sequence	Control Code	Description	Escape Sequence	Description
\a	BEL	Bell character (alert)	\\	Backslash (\) character
\b	BS	Backspace	\'	Single quotation
\f	FF	Page break/page feed (clear)	\"	Double quotation
\n	LF	Line field/line feed/return		
\r	CR	Carriage return/return to beginning of line		
\t	HT	Horizontal tab		
\v	VT	Vertical tab		

* For command line input, “2.2.7 JSON command interface” enables input and output by JSON.
Escape sequences handled by JSON shall follow JSON.

Entering a string literal on the command line is the direction for entering character string of argument values for a command. Therefore, the escape sequence in input string literal is converted to the control code in the command execution process. The character string obtained by removing the double quotation marks ('"') from string literal is used as the argument value character string

<Enter a string literal enclosed in double quotation marks ('"') as one argument>

```
> <command> "abc d\tef,g\"hi" ...[argument]<new line>
```

If the character string " abc d \ tef, g \ "hi" is entered in the command line input above, the command processes the character string "abc d <tab character> ef, g" hi ", which is a converted string literal as character string argument of the command.

2.2.3 Data transmission command

This is a data transmission command from EW-WSN, and is outputted when notifying the user of data received from the opposite Wi-SUN module.
Enter the data transmission command in the following format.

```
<data transmission command> [argument],[argument]...[argument]<new line>
```

The data transmission command is completed when <line feed> is entered.

To send a binary code that cannot be written as text in the data to be sent, use “3.3.6(1) udps” and “3.3.6(3) tcps” commands.
The number specified in the argument is specified in hexadecimal unless otherwise specified.
Refer to the explanation of data transmission command for the details. (“3.3.6(1) udps” / “3.3.6(3) tcps”)

To send the data to be sent in text, use “3.3.6(2) udpst” command.
The text character string specified in “3.3.6(2) udpst” command argument shall follow “2.2.2 Command-line”.

2.2.4 Address

When specifying an address in the argument of command or data output, specify it in MAC address or IPv6 address.

Ex. 1	Unicast (MAC address)	`aabbccddeeff0001`
	Unicast (IPv6 address)	`2001:db8::1`
Ex. 2	Multicast (MAC address)	`ffffffffffffffff`
	Multicast (IPv6 address)	`ff02::1`

2.2.5 IPv6 multicast address

The operation of IPv6 multicast address is confirmed only for the following addresses.

Link local scope	`ff02::1`	All peripheral terminals
	`ff02::2`	All peripheral routers
Realm local scope	`ff03::1`	All network terminals
	`ff03::2`	All network routers

2.2.6 Parameter commands and parameter areas

In this command application, the "parameter" function is implemented as a function to save the current set values of various commands in the non-volatile area and then sets them to the various saved set values upon restart.

With this parameter function, various values saved in the non-volatile area are referred to as "parameter area".

Refer to “3.2.8 Parameter commands” for the commands related to parameter function and set values of parameter area.

2.2.7 JSON command interface

This command application is based on the data exchange format by JSON (http://www.json.org/json-ja.html) .
Command can be inputted and outputted.

When the command input character string in JSON format is entered, the response result of command will be outputted in JSON format.

Enter the command in JSON character string in the following format.

```
> { "<command name>" : { "argument name" : argument value, "argument name": argument value, ... } }<new l
ine>
```

Enter the JSON character string of command in one line per command. (A newline character cannot be inserted in JSON character string in one command.)
The JSON character string command is executed when <newline> is entered.

Regarding the above “argument name", the argument name corresponding to each command is described in “3 Commands”.
The output of each message as JSON character string can be set by using the “3.3.1(6) json” command.

In various commands of “3 Commands”, if the command information should be obtained or the setting status in JSON format should be displayed, enter the value for the above command name in JSON format as NULL value to display and output the setting status of each command.
To obtain the command information or display the setting status, enter the command in the following format.

```
> { "<command name>" : null }<new line>
```

In this manual, the command character string in JSON format according to the above format is described as "JSON command".
Refer to the description for each JSON using “3 Commands” for the details on various commands using JSON commands.

2.2.8 920MHz (Sub-GHz) radio frequency setting

EW-WSN-FAN sets the following communication channel standard settings based on ARIB STD-T108 to 920MHz (Sub-GHz) wireless RF module.

Bandwidth : 200KHz(1ch bundle), Data rate: 50Kbps, Center frequency: 922.4MHz-928.0 MHz), Channel used: 33 - 61
Bandwidth : 400KHz(2ch bundle), Data rate:150Kbps, Center frequency: 922.5MHz-927.8 MHz), Channel used: 33 - 59
Bandwidth : 600KHz(3ch bundle), Data rate:300Kbps, Center frequency: 922.6MHz-927.4 MHz), Channel used: 33 - 57

Use “3.3.2(3) chrate” command to set the data rate (communication symbol rate).
Use “3.3.2(4) chan” command to set the channel (center frequency).

Center Frequency (MHz)	Channel No. ARIB 50 Kbps 1CH bundle	ARIB 100/150 Kbps 2CH bundle	ARIB 300 Kbps 3CH bundle
922.4	33		
922.5		33	
922.6	34		33
922.7			
922.8	35		
922.9		35	
923.0	36		
923.1			
923.2	37		
923.3		37	36
923.4	38		
923.5			
923.6	39		
923.7		39	
923.8	40		
923.9			
924.0	41		39
924.1		41	
924.2	42		
924.3			
924.4	43		
924.5		43	
924.6	44		42
924.7			
924.8	45		
924.9		45	
925.0	46		45
925.1			

Center Frequency	Channel No. ARIB 50 Kbps 1CH bundle	ARIB 100/150 Kbps 2CH bundle	ARIB 300 Kbps 3CH bundle
925.2	47	45	
925.3			45
925.4	48	47	
925.5			
925.6	49		
925.7			
925.8	50	49	48
925.9			
926.0	51		
926.1			
926.2	52	51	
926.3			
926.4	53		51
926.5			
926.6	54	53	
926.7			
926.8	55		
926.9			
927.0	56	55	54
927.1			
927.2	57		
927.3			
927.4	58	57	
927.5			
927.6	59		57
927.7			
927.8	60	59	
927.9			
928.0	61		
928.1			

3. Commands

Use "Command-line" and "JSON Command" to describe the various commands.
Describe the command string with non-format command and command arguments according to ”2.2.2 Command-line” as “command-line".
Describe the command string in JSON format according to “2.2.7 JSON command interface” as "JSON Command".

3.1 Classifications of commands

Various commands are listed below as classification.

No	Classification	Description
1	Basic commands	Classified in the basic command group such as help and firmware version display.
2	Connection setting display commands	Classified in the command group for checking the connection settings and connection set values.
3	Border Router control connection setting display commands	Classified in the command group for checking the connection settings and connection set values of Border Router.
4	Connection status display commands	Classified in the command group that displays the system status such as connection set value and connection status value, connection status, and various protocol status.
5	Border Router control connection status display commands	Classified in the command group for displaying the system status, connection status, and various protocol status such as Border Router connection set value and connection status value.
6	Data transmission commands	Classified in the data transmission command and the terminal response confirmation command group such as ping.
7	Data transmission/reception setting commands	Classified in the group of commands related to data transmission command, reception data output content setting, and packet format setting related to data transmission/reception.
8	Parameter commands	This displays and sets static values for non-volatile areas such as FlashROM.
9	Remote commands	This sends command statement to a remote terminal connected to the network that implements this command application, the command is executed on the received remote terminal, and the command execution result is returned.

3.2 List of commands

3.2.1 Basic commands

No	Command	Description
1	help	To display the help message
2	vers	To display the firmware version
3	vernum	To display the firmware version
4	reset	To reboot the Wi-SUN module
5	echo	Echo back settings
6	json	To set/display the JSON character string input and output
7	baud	To set/display the serial communication speed

3.2.2 Connection setting display commands

No	Command	Description
1	mode	To set/display the operation mode
2	auth	To set/display the authentication security behavior
3	chrate	To set/display the transmission rate
4	chan	To set/display the start and end channel
5	rccal	To set/display the RSSI threshold value in CCA operation of PHY
6	mac	To display mac address
7	macf	To set/display the incoming MAC address filtering
8	mtxctl	To set/display the MAC packet transmission total time control
9	pan	To set/display the PAN ID
10	netname	To set/display the network ID
11	ip	To set/display the IPv6 address
12	init	Terminal startup
13	atstart	Terminal autostart settings
14	tcpcon	To manually connect the TCP-Connection
15	tcpdis	To manually disconnect the TCP-Connection
16	rantsw	To set/display the antenna used by PHY
17	rpambl	To set/display the transmission preamble length to PHY(RADIO)
18	rfec	To set/display the FEC function operation of PHY(RADIO)
19	sleep	To execute sleep/awake

3.2.3 Border Router control connection setting display commands

No	Command	Description
1	leaseip	To set/display the DHCPv6 server payout fixed IP address
2	leaserng	To set/display range of payout IP address of the DHCPv6 server
3	nodef	To set/display the MAC Address Filtering for Connection Allowed Nodes
4	fnode	To check connected nodes

3.2.4 Connection status display commands

No	Command	Description
1	stat	To display the terminal status
2	rstat	To display the physical layer(PHY-RADIO) statistics
3	mstat	To display the MAC statistics
4	fstat	To display the FAN(MAC-FAN) information
5	chconfig	To display the frequency area used and transmission rate
6	chcur	To display the currently used channel number
7	fmseckey	To display the FAN MAC security key
8	nebr	To display the nearby terminal information
9	parent	To display the parent terminal information
10	rplinf	To display the information about RPL
11	tcpstat	To display the TCP status

3.2.5 Border Router control connection status display commands

No	Command	Description
1	rplsr	To display the source routing table information configured by RPL
2	leased	To display the IPv6 address currently issued by DHCPv6 server

3.2.6 Data transmission commands

No	Command	Description
1	udps	UDP data transmission(binary code transmission)
2	udpst	UDP data transmission(text character string transmission)
3	tcps	TCP data transmission
4	ping	ping execution

3.2.7 Data transmission/reception setting commands

No	Command	Description
1	udpopts	To display the UDP data transmission/reception setting
2	tcpopts	To display the TCP data transmission/reception setting

3.2.8 Parameter commands

No	Command	Description
1	param	Setting display
2	save	Save settings
3	clear	Clear settings
4	svrst	Reset after saving settings
5	clrst	Reset after clearing settings

3.2.9 Remote commands

No	Command	Description
1	rmtcmd	Remote command(command execution on remote terminal)
2	rmtopts	To set the display for remote commands(command execution on remote terminal)

3.3 Command specifications

3.3.1 Basic commands

(1) help

Overview

This command displays the help message for each command.

Command-line format

```
help
```

JSON command format

```
{ "help" : null }
```

Command argument

Nothing

Command-line display format

The help message is displayed in the following format with values described in **Display format contents** below.

```
<fmt> - <desc>
```

JSON command display format

The help message is displayed in the following format with values described in **Display format contents** below.

```
{ "help" : { "fmt" : <string>, "desc" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
fmt	string	Command format	To display the command name and argument format.
desc	string	Command overview	To display the summary of command operation.

Example of command line usage

```
>help
  help                  - disp help
  vers                  - disp firmware version string
  vernum                - disp firmware version number
    ~~~ Omitted ~~~
  save                  - save parameters
  clear                 - clear parameters
  svrst [reset_delay_sec] - save parameters and reset
  clrst [reset_delay_sec] - clear parameters and reset
>
```

Example of JSON command usage

```
>{ "help" : null }
{ "help" : { "fmt" : "help", "desc" : "disp help" } }
{ "help" : { "fmt" : "vers", "desc" : "disp firmware version string" } }
{ "help" : { "fmt" : "vernum", "desc" : "disp firmware version number" } }
    ~~~ Omitted ~~~
{ "help" : { "fmt" : "save", "desc" : "save parameters" } }
{ "help" : { "fmt" : "clear", "desc" : "clear parameters" } }
{ "help" : { "fmt" : "svrst [reset_delay_sec]", "desc" : "save parameters and reset" } }
{ "help" : { "fmt" : "clrst [reset_delay_sec]", "desc" : "clear parameters and reset" } }
>
```

(2) vers

Overview

This command displays the firmware version.

Command-line format

```
vers
```

JSON command format

```
{ "vers" : null }
```

Command argument

Nothing

Command-line display format

The firmware version is displayed in the following format with values described in **Display format contents** below.

```
////////////////////////////////////////////////////////
// <Copyright>
// <FirmwareBSP>
// <ProfileBuild>
////////////////////////////////////////////////////////
```

JSON command display format

The firmware version is displayed in the following format with values described in **Display format contents** below.

```
{ "vers" : { "Copyright" : <string>, "FirmwareBSP" : <string>, "ProfileBuild" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
Copyright	string	Copyright	To display the copyright.
FirmwareBSP	string	Firmware version BSP name, board name	To display the following: EW-WSN-FAN Firmware Version BSP name Board name(board information)
ProfileBuild	string	Wi-SUN support profile name Module build date	To display the following: Wi-SUN support profile name Module build date

Example of command line usage

```
>vers
////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Mar 11 2020 10:14:03)
////////////////////////////////////////////////////////
>
```

Example of JSON command usage

```
>{ "vers" : null }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Mar 11 2020 10:14:03)" } }
>
```

(3) vernum

Overview

This command displays the firmware version. (Simple display of "3.3.1(2) vers" commands)

Command-line format

```
vernum
```

JSON command format

```
{ "vernum" : null }
```

Command argument

Nothing

Command-line display format

The current operation profile mode is displayed in the following format with values described in **Display format contents** below.

```
vernum <version>
```

JSON command display format

The current operation profile mode is displayed in the following format with values described in **Display format contents** below.

```
{ "vernum" : "<version>"  }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
version	string	Firmware version string	To display the following: EW-WSN-FAN Firmware Version Board name Module build date

Example of command line usage

```
>vernum
vernum EW-WSN-FAN-1.0.56.60 linux (build Mar 11 2020 10:14:03)
>
```

Example of JSON command usage

```
>{ "vernum" : null }
{ "vernum" : "EW-WSN-FAN-1.0.56.60 linux (build Mar 11 2020 10:14:03)" }
>
```

(4) reset

Overview

This command resets (reboots) the system of Wi-SUN module.
This displays the execution delay time (in seconds) when the command is executed.

Command-line format

```
reset [delay_sec]
```

JSON command format

```
{ "reset" : null }
{ "reset" : <number> }
{ "reset" : { "delay_sec" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Display Value	Description
delay_sec	number	0~3600	System reset execution delay time(in seconds)

If the argument is omitted in the above command-line format, the system reset will be executed immediately with delay time of 0(seconds).
If the value of command name is set to null in JSON command, the system reset is executed immediately with delay time of 0(seconds).

Command-line display format

The system reset execution delay time (in seconds) is displayed in the following format with values described in **Display format contents** below.

```
reset delay <delay_sec>sec
```

JSON command display format

The system reset execution delay time (in seconds) is displayed in the following format with values described in **Display format contents** below.

```
{ "reset" : <delay_sec> }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
delay_sec	number	0 or more (Input value)	System reset execution delay time(in seconds)

Example of command line usage

```
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>reset 5
reset delay 5sec
>[~~~ 5 seconds later ~~~]
inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
```

Example of JSON command usage

```
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
>{ "reset" : 5 }
{ "reset" : 5 }
>[~~~ 5 seconds later ~~~]
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
```

(5) echo

Overview

This command sets the echoback.
It displays the current echoback settings.

Command-line format

```
echo [enable]
```

JSON command format

```
{ "echo" : null }
{ "echo" : <bool> }
{ "echo" : { "enable" : <bool> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	JSON Command Set Value	Initial Value	Description
enable	bool	0/off	0/false		To disable echo
		Numbers other than 0/on	Numbers other than 0/true	✓	To enable echo

If the argument is omitted, the current echoback setting is displayed.
There will be no prompt when echo is disabled.
If the value of command name is set to null in JSON command, the operation setting of JSON character string output is displayed.

Command-line display format

The current echoback settings are displayed similar to the **Command-line format** above.

JSON command display format

The current echoback settings are displayed in JSON format similar to the **JSON command format** above.

Example of command line usage

```
>echo
echo on
>echo 0
(Input "echo")
echo off
(Input "echo 1")
echo on
>
```

Example of JSON command usage

```
>{ "echo" : null }
{ "echo" : true }
>{ "echo" : false }
(Input "{ "echo" : null }")
{ "echo" : false }
(Input "{ "echo" : true }")
{ "echo" : true }
>
```

(6) json

Overview

This command sets the operation of JSON character string output.
It displays the operation settings of the current JSON character string output.

By turning on (enable) the operation setting in the JSON character string, the notification message from EW-WSN-FAN protocol stack and "4 Message Output" are outputted as JSON character string.
(Regarding the response result of command, only the command input in JSON character string will be the result output with the JSON character string.)

Command-line format

```
json [enable]
```

JSON command format

```
{ "json" : null }
{ "json" : <bool> }
{ "json" : { "enable" : <bool> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	JSON Command Set Value	Initial Value	Description
enable	bool	0/off	0/false	✓	To disable JSON output. This outputs the notification message from protocol stack and notification message by "4 Message Output" as text message that is not a JSON character string.
		Numbers other than 0/on	Numbers other than 0/true		To enable JSON output. This outputs the notification message from protocol stack and notification message by "4 Message Output" as JSON character string.

If the argument is omitted on the command-line, the current operation setting of JSON character string output is displayed.
If the value of command name is set to null in JSON command, the operation setting of current JSON character string output is displayed.

Command-line display format

The current JSON character string output operation settings are displayed similar to the **Command-line format** above.

JSON command display format

The current JSON character string output operation settings are displayed similar to the **Jason command format** above.

Example of command line usage

```
>clear
clear parameter is cleared
>json
json off
>
>json 1
json on
>
>save
save parameter is saved
>reset
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Mar 11 2020 10:14:03)" } }
{ "init" : "0(NONE)" }
>
```

Example of JSON command usage

```
>{ "clear" : null }
{ "clear" : { "desc" : "parameter is cleared" } }
>{ "reset" : null }
inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Mar 11 2020 10:14:03)
////////////////////////////////////////////////////////////
init 0(NONE)
>{ "json" : null }
{ "json" : false }
>
>{ "json" : true }
{ "json" : true }
>
>{ "save" : null }
{ "save" : { "desc" : "parameter is saved" } }
>{ "reset" : null }
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Mar 11 2020 10:14:03)" } }
{ "init" : "0(NONE)" }
>
```

(7) baud

Overview

This command sets the serial communication speed with command application.
It displays the current serial communication speed setting.
The serial communication speed for which operation is checked depends on the following platforms.

Platform (BSP)	2400	4800	9600	14400	19200	38400	57600	115200	230400	460800
BP35C4(ROHM)	✓	✓	✓	✓	✓	✓	✓	✓	-	-
BP35C5(ROHM)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Command-line format

```
baud [rate]
```

JSON command format

```
{ "baud" : null }
{ "baud" : <number> }
{ "baud" : { "rate" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	Initial Value	Description
rate	number	2400/4800/9600/14400/19200/ 38400/57600/115200/230400/460800	115200	Serial communication speed with command application

If the argument is omitted, the current serial communication speed setting is displayed.
If the value of command name is set to null in JSON command, the current serial communication speed setting is displayed.

Command-line display format

The current serial communication speed settings are displayed similar to the **Command-line format** above.

JSON command display format

The current serial communication speed settings are displayed in JSON format similar to the **JSON command format** above.

Example of command line usage

```
>baud
baud 115200
>baud 230400
(Changed the terminal serial communication speed setting to 230400)
>
>baud
>baud 230400
>
```

Example of JSON command usage

```
>{ "baud" : null }
{ "baud" : 115200 }
>{ "baud" : 230400 }
(Changed the terminal serial communication speed setting to 230400)
>
>{ "baud" : null }
{ "baud" : 230400 }
>
```

3.3.2 Connection setting display commands

(1) mode

Overview

This command sets the operation profile mode.
It displays the current operating profile mode.

Command-line format

```
mode <profile>
```

JSON command format

```
{ "mode" : null }
{ "mode" : <string> }
{ "mode" : { "profile" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set value	Initial Value	Description
profile	string	"0" or "NONE"		Non-operation mode
		"1" or "FAN"	✓	FAN profile mode

If the argument is omitted, the current operation profile mode is displayed below.
If the value of command name is set to null in JSON command, the current operation profile mode is displayed.

Command-line display format

The values described in **Display format contents** below display the current operation profile mode in the following format.

```
mode <profile (name) >
```

JSON command display format

The values described in **Display format contents** below display the current operation profile mode in the following format.

```
{ "mode" : <profile(name) > }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
profile(name)	string	profile: 0 or 1	To display the number corresponding to profile mode name by the name below. 0: Non-operation mode 1: FAN profile mode
		name: "NONE" or "FAN"	To display the profile mode name corresponding to the above profile number. "NONE": Non-operation mode "FAN": FAN profile mode

Example of command line usage

```
>mode
mode 1(FAN)
>mode 0
mode 0(NONE)
>mode
mode 0(NONE)
>mode 1
mode 1(FAN)
>mode
mode 1(FAN)
>mode NONE
mode 0(NONE)
>mode
mode 0(NONE)
>mode FAN
mode 1(FAN)
>mode
mode 1(FAN)
>
```

Example of JSON command usage

```
>{ "mode" : null }
{ "mode" : "1(FAN)" }
>{ "mode" : "0" }
{ "mode" : "0(NONE)" }
>{ "mode" : null }
{ "mode" : "0(NONE)" }
>{ "mode" : "1" }
{ "mode" : "1(FAN)" }
>{ "mode" : null }
{ "mode" : "1(FAN)" }
>{ "mode" : "NONE" }
{ "mode" : "0(NONE)" }
>{ "mode" : null }
{ "mode" : "0(NONE)" }
>{ "mode" : "FAN" }
{ "mode" : "1(FAN)" }
>{ "mode" : null }
{ "mode" : "1(FAN)" }
>
```

(2) auth

Overview

This command sets the FAN authentication connection and authentication security operation.
It displays the current operating mode for FAN authentication connection and authentication security operation.

Command-line format

```
auth [num]
```

JSON command format

```
{ "auth" : null }
{ "auth" : <number> }
{ "auth" : { "num" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set value	Initial Value	Description
num	number	0		Unauthenticated security mode
		1	✓	Authenticated security mode EAPOL over IEEE802.15.4E defined in Wi-SUN-FAN standard is connected by TLS authentication operation by RADIUS. For IEEE802.15.4E packets, encrypted packet switching operation is performed using the security key exchanged in this authentication operation.

If the argument is omitted, the current authentication security operation settings are displayed below.
If the value of command name is set to null in JSON command, the current operation mode is displayed.

Command-line display format

The current operation mode set value for the authentication security operation is displayed similar to the **Command-line format** above.

JSON command display format

The current operation mode set value for authentication security operation is displayed in JSON format similar to the **JSON command format** above.

Example of command line usage

```
>auth
auth 1
>auth 0
auth 0
>auth
auth 0
>auth 1
auth 1
>
```

Example of JSON command usage

```
>{ "auth" : null }
{ "auth" : 1 }
>{ "auth" : 0 }
{ "auth" : 0 }
>{ "auth" : null }
{ "auth" : 0 }
>{ "auth" : { "num" : 1 } }
{ "auth" : 1 }
>
```

(3) chrate

Overview

This command sets the PHY transmission rate.
It displays the current set value for the PHY transmission rate.
The configurable PHY transmission rate is platform dependent.

Platform (KBps)	50	100	150	300
BP35C4(ROHM)	✓	✓	✓	-
BP35C5(ROHM)	✓	✓	✓	✓

Command-line format

```
chrate [rate]
```

JSON command format

```
{ "chrate" : null }
{ "chrate" : <number> }
{ "chrate" : { "rate" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set value	Initial Value	Description
rate	number	50/100 /150/ 300	150	To specify the transmission rate of tPHY to be used in decimal numbers in Kbps. (Range: 50 or 100 or 150 or 300) * The transmission rate that can be set depends on the hardware used.

After changing the set value of rate (PHY transmission rate) by this command, the PHY start and end channel in “3.3.2(4) chan” is reset to default value.
If the FAN connection status is not "INIT"(connection status number 0: initial status) (communication enabled status during connection or connection with other node terminals), the setting of the above rate (PHY transmission rate) is not changed to current value, but is changed to parameter area.
After setting this command, execute “3.3.8(2) save” and “3.3.8(4) svrst” commands to save the parameters. After restarting, it will automatically start at the above rate (PHY transmission rate) set by this command. (Refer to “3.3.4(4) fstat” for the FAN connection status.)
If the argument is omitted, the current set value for the PHY transmission rate is displayed.
If the value of command name is set to null in JSON command, the current set value for the PHY transmission rate is displayed.

Command-line display format

The values described in **Display format contents** below are displayed in the following format as set values for the current channel.

```
chrate <rate>Kbps (prm) :<prm_rate>Kbps
```

JSON command display format

The values described in **Display format contents** below are displayed in the following format as set values for the current channel.

```
{ "chrate" : { "rate" : <number>, "prm_rate" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
rate	number	50/100/150/ 300	To display the current set value of PHY transmission rate in Kbps.
prm_rate	number	50/100/150/ 300	To display the PHY transmission rate set value in parameter area in Kbps. (The set value after restart is displayed.)

Example of command line usage

```
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chrate
chrate 150Kbps (prm):150Kbps
>
>chrate 50
chrate 50Kbps (prm):50Kbps
>
>save
save parameter is saved
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chrate
chrate 50Kbps (prm):50Kbps
>
>init 1
init 1(BORDER)
inf 2b,62,0,5      {   FMng: changed fan join state (0 -> 5)     }
>
>chrate 300
chrate 50Kbps (prm):300Kbps
>
>svrst
svrst parameter is saved and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chrate
chrate 300Kbps (prm):300Kbps
>
```

Example of JSON command usage

```
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chrate" : null }
{ "chrate" : { "rate" : 150, "prm_rate" : 150 } }
>
>{ "chrate" : 50 }
{ "chrate" : { "rate" : 50, "prm_rate" : 50 } }
>
>{ "save" : null }
{ "save" : { "desc" : "parameter is saved" } }
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chrate" : null }
{ "chrate" : { "rate" : 50, "prm_rate" : 50 } }
>
>{ "init" : "BORDER" }
{ "init" : "1(BORDER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 5 } }
>
>{ "chrate" : { "rate" : 300} }
{ "chrate" : { "rate" : 50, "prm_rate" : 300 } }
>
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chrate" : null }
{ "chrate" : { "rate" : 300, "prm_rate" : 300 } }
>
```

(4) chan

Overview

This command sets the start and end channel for PHY.
It displays the current settings for PHY start and end channels.

Command-line format

```
chan [low] [high]
```

JSON command format

```
{ "chan" : null }
{ "chan" : <number> }
{ "chan" : { "low" : <number>, "high" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	JSON Command Set Value	Initial Value	Description
low	number	33~61		33	To specify the start channel number in decimal. If the set channel number is not the channel start number in the corresponding channel number area, it is set to the start number of corresponding channel.
high	number	33~61		59	To specify the end channel number in decimal. If the set channel number is not the channel end number in the corresponding channel number area, it is set to the end number of the corresponding channel. If this field is omitted, the high=low channel will be set as fixed channel.

Refer to “2.2.8 920MHz (Sub-GHz) radio frequency setting” for the channel number and center frequency that can be set.

After changing the PHY transmission rate set value using “3.3.2(3) chrate” command, the above low/high (PHY start and end channel) will be reset to default value after.

If the FAN connection status is not "INIT" (0: initial status) (This is the connecting or communicable status that can connect to other node terminals), the setting of low/high (PHY start and end use channel) is not changed to current value, but to parameter area.

After setting this command, execute “3.3.8(2) save” and “3.3.8(4) svrst” commands to save the parameters. Then after restarting, it will automatically start at the above low/high (PHY start and end use channel) with the set value of this command. (Refer to “3.3.4(4) fstat” for the FAN connection status.)

If the argument is omitted, the set value for current channel is displayed.
If the value of command name is set to null in JSON command, the current set value for the channel including PHY start and end channel is displayed.

Command-line display format

The values described in **Display format contents** below are displayed in the following format as set values for the current channel.

```
chan low=<low><=>high=<high>,num=<num> (prm) : low=<prm_low><=>high=<prm_high>,num=<prm_num>
```

JSON command display format

The values described in **Display format contents** below are displayed in the following format as set values for the current channel.

```
{ "chan" : { "low" : <number>, "high" : <number>, "num" : <number>, "prm_low" : <number>, "prm_high" : <n
umber>, "prm_num" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
low	number	0~128	To display the current set value of start channel number.
high	number	0~128	To display the currently set value of end channel number.
num	number	0~128	To display the current set value of number of channels used.
prm_low	number	0~128	To display the use start channel number of parameter area. (The set value after restart is displayed.)
prm_high	number	0~128	To display the end channel number of parameter area. (The set value after restart is displayed.)
prm_num	number	0~128	To display the number of channels used in parameter area. (The set value after restart is displayed.)

Example of command line usage

```
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chrate
chrate 150Kbps (prm):150Kbps
>
>chan
chan low=33<=>high=59,num=14 (prm):low=33<=>high=59,num=14
>
>chrate 50
chrate 50Kbps (prm):50Kbps
>
>chan
chan low=33<=>high=61,num=29 (prm):low=33<=>high=61,num=29
>
>chan 45
chan low=45<=>high=45,num=1 (prm):low=45<=>high=45,num=1
>
>save
save parameter is saved
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chan
chan low=45<=>high=45,num=1 (prm):low=45<=>high=45,num=1
>
>init 1
init 1(BORDER)
inf 2b,62,0,5      {   FMng: changed fan join state (0 -> 5)     }
>
>chan 35 57
chan low=45<=>high=45,num=1 (prm):low=35<=>high=57,num=23
>
>svrst
svrst parameter is saved and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>chan
chan low=35<=>high=57,num=23 (prm):low=35<=>high=57,num=23
>
```

Example of JSON command usage

```
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chrate" : null }
{ "chrate" : { "rate" : 150, "prm_rate" : 150 } }
>
>{ "chan" : null }
{ "chan" : { "low" : 33, "high" : 59, "num" : 14, "prm_low" : 33, "prm_high" : 59, "prm_num" : 14 } }
>
>{ "chrate" : 50 }
{ "chrate" : { "rate" : 50, "prm_rate" : 50 } }
>
>{ "chan" : null }
{ "chan" : { "low" : 33, "high" : 61, "num" : 29, "prm_low" : 33, "prm_high" : 61, "prm_num" : 29 } }
>
>{ "chan" : 45 }
{ "chan" : { "low" : 45, "high" : 45, "num" : 1, "prm_low" : 45, "prm_high" : 45, "prm_num" : 1 } }
>
>{ "save" : null }
{ "save" : { "desc" : "parameter is saved" } }
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chan" : null }
{ "chan" : { "low" : 45, "high" : 45, "num" : 1, "prm_low" : 45, "prm_high" : 45, "prm_num" : 1 } }
>
>{ "init" : "BORDER" }
{ "init" : "1(BORDER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 5 } }
>
>{ "chan" : { "low" : 35, "high" : 57 } }
{ "chan" : { "low" : 45, "high" : 45, "num" : 1, "prm_low" : 35, "prm_high" : 57, "prm_num" : 23 } }
>
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 13:57:36)" } }
{ "init" : "0(NONE)" }
>
>{ "chan" : null }
{ "chan" : { "low" : 35, "high" : 57, "num" : 23, "prm_low" : 35, "prm_high" : 57, "prm_num" : 23 } }
>
```

(5) rccal

Overview

This command sets the RSSI threshold for CCA operation of PHY.
It displays the current RSSI threshold for CCA (Clear Channel Assessment) operation of PHY (RADIO).

Command-line format

```
rccal <rssi>
```

JSON command format

```
{ "rccal" : null }
{ "rccal" : <number> }
{ "rccal" : { "rssi" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
rssi	number	-128 to initial value	Wi-SUN platform dependent	To specify the RSSI threshold (-128 to initial value) for CCA (Clear Channel Assesment) operation of PHY (RADIO). It is not possible to set a value higher than the initial value.

Platform	Initial Value
BP35C4(ROHM)	-90
BP35C5(ROHM)	-84

If the argument is omitted, the current RSSI threshold is displayed.
If the value of command name is null in JSON command, the RSSI threshold is displayed.

Command-line display format

The RSSI threshold is displayed similar to the **Command-line format** above.

JSON command display format

The RSSI threshold is displayed similar to the **JSON command format** above.

Example of command line usage

```
>rccal
rccal -90
>
>rccal -91
rccal -91
>
```

Example of JSON command usage

```
>{ "rccal" : null }
{ "rccal" : -90 }
>
>{ "rccal" : -91 }
{ "rccal" : -91 }
>
>{ "rccal" : { "rssi" : -90 } }
{ "rccal" : -90 }
>
```

(6) mac

Overview

This command displays the mac address.

Command-line format

```
mac
```

JSON command format

```
{ "mac" : null }
```

Command argument

Nothing

Command-line display format

The mac address is displayed in the following format with values described in the **Display format contents** below.

```
mac <MAC64B>,LLA<IP128BLLA>
```

JSON command display format

The mac address is displayed in the following format with values described in the **Display format contents** below.

```
{ "mac" : { "MAC64B" : <string>, "IP128BLLA" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Set Value	Description
MAC64B	string	<EUI64MAC address>	To display the MAC address in EUI64 format.
IP128BLLA	string	<IPv6 link-local address>	To display the IPv6 link-local address generated by the MAC address.

Example of command line usage

```
>mac
mac <0001020304050601>,LLA<fe80::201:203:405:601>
>
```

Example of JSON command usage

```
>{ "mac" : null }
{ "mac" : { "MAC64B" : "0001020304050601", "IP128BLLA" : "fe80::201:203:405:601" } }
>
```

(7) macf

Overview

This command sets the received packet filtering in MAC layer. (The received packet is filtered by MAC address.)
It displays the received packet filtering settings in the current MAC layer.
The number of MAC addresses that can be registered depends on the platform.

Platform	Settable number
BP35C4(ROHM)	128
BP35C5(ROHM)	128

Command-line format

```
macf [role] [MAC64B]
```

JSON command format

```
{ "macf" : null }
{ "macf" : <string> }
{ "macf" : { "role" : <string> } }
{ "macf" : { "role" : <string>, "MAC64B" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial value	Description
role	string	"allow"	✓	This is a permission setting. To receive the received packets with MAC address specified by the argument “MAC” without filtering by MAC layer. If the argument “MAC64B” is omitted, the default rule is changed to allow.
		"deny"		This is a rejection setting. To filter by MAC layer the received packets with the source MAC address specified by the argument “MAC” and deny. If the argument “MAC64B” is omitted, the default rule is changed to deny.
		"del"		Delete rule. To delete the rule setting for MAC address specified by the argument “MAC64B”. Filtering the incoming packets for deleted MAC addresses follows the default rules.
		"clr"		Delete all rules. To delete all currently set MAC address rule settings. The default rules are not changed. Filtering the incoming packets for deleted MAC addresses follows the default rules.
MAC64B	string	<EUI64MAC address>	-	To specify the EUI64 format MAC address to be operated with the argument “role”.

If the argument is omitted, the current rule settings are displayed below.
If the value of command name is set to null in JSON command, the current rule settings will be displayed below.

Command line display format

The received packet filtering settings in the current MAC layer is displayed in the following format with values described in **Display format contents** below.

```
macf num:<num>, max:<max>
macf default ( <role>  )
macf <MAC64B> ( <role> )
macf <MAC64B> ( <role> )
…(After that, the number of registered units is displayed.)
```

JSON command display format

The received packet filtering settings in the current MAC layer is displayed in the following format with values described in **Display format contents** below.

```
{ "macf" : { "num" : <number>, "max" : <number> } }
{ "macf" : { "default" : <string> } }
{ "macf" : { "role" : <string>, "MAC64B" : <string> } }
{ "macf" : { "role" : <string>, "MAC64B" : <string> } }
…(After that, the number of registered units is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0 or more	To display the current setting number of registered node terminals
max	number	Platform dependent	To display the maximum number of units that can be registered on the current operating platform.
default	string	"allow"	This is a permission setting. To receive the received packets without filtering on MAC layer. All received packets other than "deny" (= set to deny) in “role” of “MAC64B” below are received without filtering.
		"deny"	This is a rejection setting. To filter the received packets by MAC layer and deny. All received packets other than "allow" (= permission is set) in “role” of “MAC64B” below are filtered by the MAC layer and denied.
MAC64B	string	<EUI64MAC address>	The terminal MAC address to which the following “role” is applied.
role	string	"allow"	This is a permission setting. To receive the packets with the above “MAC64B” address without filtering by MAC layer.
		"deny"	This is a rejection setting. The received packets with the above “MAC64B” address are filtered by MAC layer and not received.

Example of command line usage

<To allow all packets from MAC addresses not set in the rule>

```
>macf allow
```

<To deny all packets from MAC addresses not set in the rule>

```
>macf deny
```

<To deny packets from the 0011223344556677>

```
>macf deny 0011223344556677
```

<To allow packets from the 0011223344556677>

```
>macf allow 0011223344556677
```

<To show the current rule setting>

```
>macf
macf num:2, max:64
macf default           ( allow )
macf <0011223344556677> ( deny  )
macf <0011223344556688> ( allow )
```

<To delete the rule of 0011223344556677>

```
>macf del 0011223344556677
```

<To delete all registered rules>

```
>macf clr
```

Example of JSON command usage

<To allow all packets from MAC addresses not set in the rule>

```
>{ "macf" : "allow" }
```

<To deny all packets from MAC addresses not set in the rule>

```
>{ "macf" : { "role" : "deny"} }
```

<To deny packets from the 0011223344556677>

```
>{ "macf" : { "role" : "deny",  "MAC64B" : "0011223344556677" } }
```

<To allow packets from the 0011223344556677>

```
>{ "macf" : { "role" : "allow", "MAC64B" : "0011223344556688" } }
```

<To show the current rule setting ＞

```
>{ "macf" : null }
{ "macf" : { "num" : 2, "max" : 64 } }
{ "macf" : { "default" : "allow" } }
{ "macf" : { "role" : "deny", "MAC64B" : "0011223344556677" } }
{ "macf" : { "role" : "allow", "MAC64B" : "0011223344556688" } }
```

<To delete the rule of 0011223344556677>

```
>{ "macf" : { "role" : "del",  "MAC64B" : "0011223344556677" } }
```

<To delete all registered rules>

```
>{ "macf" : "clr" }
```

(8) mtxctl

Overview

This command suppresses the transmission based on the total transmission time in MAC layer.
It displays the transmission suppression setting value based on the total transmission time in MAC layer.
Based on ARIB STD-T108, the total transmission time is measured in MAC layer. If the total suppression time is exceeded, the transmission of packet is canceled and discontinued before transmission.

* With this function, if the total transmission time is exceeded, the transmission of the packet is discontinued and canceled regardless of the type of transmission packet in MAC layer.

* The total suppression time is calculated by accumulating the transmission time in 30 second units, and if the estimated transmission time of packet exceeds the total suppression time, the transmission of packet is discontinued and canceled.

Command-line format

```
mtxctl [enable]
```

JSON command format

```
{ "mtxctl" : null }
{ "mtxctl" : <bool> }
{ "mtxctl" : { "enable" : <bool> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	JSON Command-line Set Value	Initial Value	Description
enable	bool	0 or off	false		(Disable): To discontinue the transmission suppression. * This cannot be set. Even if 0 is specified, it will be set to 1.
		1 or on	true	✓	(Enable): To measure the total time in MAC layer. If the transmission time of packet exceeds the suppression total time before transmission, the transmission of packet is discontinued and canceled.

If the argument is omitted, the setting of the total transmission time in current MAC layer is displayed below.
If the value of command name is set to null in JSON command, the current operation mode is displayed.

Command-line display format

The set value of transmission suppression by the total transmission time in MAC layer is displayed by on/off similar to the **Command-line format** above.

JSON command display format

The set value of transmission suppression by the total transmission time in MAC layer is displayed in JSON format similar to the **JSON command format** above.

Example of command line usage

```
>mtxctl
mtxctl on
>
>mtxctl 0
mtxctl on
>
```

Example of JSON command usage

```
>{ "mtxctl" : null }
{ "mtxctl" : true }
>
>{ "mtxctl" : false }
{ "mtxctl" : true }
>
>{ "mtxctl" : { "enable" : false} }
{ "mtxctl" : true }
>
```

(9) pan

Overview

This command displays the current PAN-ID.
It sets the PAN-ID.
* The PAN-ID setting is enabled only when used with Border Router.
(When using with Leaf/Router, PAN-ID setting is not necessary.
The PAN-ID used in Leaf / Router will be overwritten by PAN-ID distributed from Border Router when participating PAN.)

Command-line format

```
pan [PAN16B]
```

JSON command format

```
{ "pan" : null }
{ "pan" : <string> }
{ "pan" : { "PAN16B" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
PAN16B	string	<PAN-ID>	NONE(0000) (If started with Border (execute the init command) with 0000 initial value, the PAN-ID will be automatically set to cafe.)	To specify the PAN-ID.

If the argument is omitted, the current PAN-ID is displayed below.
If the value of command name is null in JSON command, the current PAN-ID is displayed.

Command-line display format

The current PAN-ID is displayed similar to the **Command-line format** above.

JSON command display format

The current PAN-ID is displayed in JSON format similar to the **JSON command format** above.

Example of command line usage

```
>pan
pan <NONE>
>
>pan
pan <cafe>
>
>pan caff
pan <caff>
>
```

Example of JSON command usage

```
>{ "pan" : null }
{ "pan" : { "PAN16B" : "NONE" } }
>
>{ "pan" : "cafe" }
{ "pan" : { "PAN16B" : "cafe" } }
>
>{ "pan" : { "PAN16B" : "caff" } }
{ "pan" : { "PAN16B" : "caff" } }
>
```

(10) netname

Overview

This command sets the FAN network ID.
(If the FAN network ID does not match with Border Router, the Border Router cannot participate in PAN connection.)
It displays the current FAN network ID setting.

Command-line format

```
netname [name]
```

JSON command format

```
{ "netname" : null }
{ "netname" : <string> }
{ "netname" : { "name" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
name	string	ASCII string (1-32 bytes)	Wi-SUN-FAN	To specify the FAN network ID. If the input character string is entered as a string literal enclosed in double quotation marks ('"'), the double quotation marks ('"') will not be included in the FAN network ID character string. Refer to “2.2.2 Command-line” for the details,

If the FAN connection status is not "INIT" (0: initial status) (This is the connecting or communicable status that can connect to other node terminals), the setting will be changed to parameter area without changing the setting of the above name (FAN network ID) to the current value.

After setting this command, execute the “3.3.8(2) save” and “3.3.8(4) svrst” commands to save the parameters. Then after restarting, the settings of this command will be used automatically and start with the above name (FAN network ID). (Refer to “3.3.4(4) fstat” for the FAN connection status.)

If the argument is omitted, the current setting value for FAN network ID is displayed.
If the value of command name is set to null in JSON command, the current setting value of FAN network ID is displayed.

Command-line display format

The values described in **Display format contents** below are displayed in the following format as set values for current channel.

```
netname <name> (prm) :<prm_name>
```

JSON command display format

The values described in **Display format contents** below are displayed in the following format as set values for current channel.

```
{ "netname" : { "name" : <number>, " prm_name" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
name	string	ASCII string (1-32 bytes)	To display the current setting value of FAN network ID. The network ID character string to be shown is displayed as a string literal enclosed in double quotation marks ('"'). (The double quotation marks ('"') of string literal to be displayed are not included in the set value of FAN network ID character string.) When displaying a character in which the FAN network ID character string to be shown contains control character represented by escape character ('\'), it converts the control character to a 2-byte character string including the escape character ('\') and displays it.
prm_name			To display the FAN network ID setting value in parameter area. (The set value after restart is displayed.)

Example of command line usage

```
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>netname
netname "Wi-SUN-FAN" (prm):"Wi-SUN-FAN"
>
>netname MYNETNAME
netname "MYNETNAME" (prm):"MYNETNAME"
>
>save
save parameter is saved
>
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>netname
netname "MYNETNAME" (prm):"MYNETNAME"
>
>init 1
init 1(BORDER)
inf 2b,62,0,5      {   FMng: changed fan join state (0 -> 5)     }
>
>netname "MY WI-SUN FAN NETNAME"
netname "MYNETNAME" (prm):"MY WI-SUN FAN NETNAME"
>
>svrst
svrst parameter is saved and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>netname
netname "MY WI-SUN FAN NETNAME" (prm):"MY WI-SUN FAN NETNAME"
>
```

Example of JSON command usage

```
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "netname" : null }
{ "netname" : { "name" : "Wi-SUN-FAN", "prm_name" : "Wi-SUN-FAN" } }
>
>{ "netname" : "MYNETNAME" }
{ "netname" : { "name" : "MYNETNAME", "prm_name" : "MYNETNAME" } }
>
>{ "save" : null }
{ "save" : { "desc" : "parameter is saved" } }
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "netname" : null }
{ "netname" : { "name" : "MYNETNAME", "prm_name" : "MYNETNAME" } }
>
>{ "init" : "BORDER" }
{ "init" : "1(BORDER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 5 } }
>
>{ "netname" : { "name" : "MY WI-SUN FAN NETNAME" } }
{ "netname" : { "name" : "MYNETNAME", "prm_name" : "MY WI-SUN FAN NETNAME" } }
>
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "netname" : null }
{ "netname" : { "name" : "MY WI-SUN FAN NETNAME", "prm_name" : "MY WI-SUN FAN NETNAME" } }
>
```

(11) ip

Overview

This command displays the current IPv6 address.
It sets the IPv6 global address.

The IPv6 global address setting is enabled only when used with Border Router.
When using in Router/Leaf terminal operation mode, the IPv6 global address setting is not necessary.
The IPv6 global address when used in Router/Leaf terminal operation mode uses the IPv6 address distributed and updated from the DHCP server when participating PAN.

* IPv6 link-local address cannot be set. (It is automatically generated based on MAC address.)

Command-line format

```
ip [IP128BGBL/prefix]
```

JSON command format

```
{ "ip" : null }
{ "ip" : <string> }
{ "ip" : { "IP128BGBL/prefix" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
IP128BGBL/prefix	string	IP128GBL : <IPv6 address>	NONE	To set the IPv6 global address. (An abbreviated IPv6 address can also be entered)
		prefix : 0~128	64	To set the prefix length of IPv6 global address.

The IPv6 global addresses that can be set are as follows.

IPv6 Unique Local Address (ULA) (Address block = fc00::/7)
IPv6 Global Unicast Address (GUA) (Address block = 2000::/3)

If the FAN connection status is not "INIT" (0: initial status) (This is the connecting or communicable status that can connect to other node terminals), the setting will be changed to parameter area without changing the setting of the above IP128B (IPv6 address) to current value.
After setting this command, execute the “3.3.8(2) save” and “3.3.8(4) svrst” commands to save the parameters. Then after restarting as Border Router, it will automatically start with the above IP128BGBL/prefix (IPv6 address and prefix length), which is the set value of this command.
When started in Router and Leaf terminal operating mode, the IPv6 address distributed and updated from DHCP server is used regardless of the IPv6 global address set in parameter area. (Refer to “3.3.4(4) fstat” for the FAN connection status.)
If the IPv6 global address in parameter area is not set when starting in Border Router terminal operation mode, it is automatically set to "2001: db8 :: 1/64".

If the argument is omitted, the current IPv6 address is displayed.
If the value of command name is null in JSON command, the current IPv6 address is displayed.

Command-line display format

The values described in **Display format contents** below are displayed in the following format as current IPv6 address set values.

```
ip LLA<IP128BLLA>,GBL< IP128BGBL/prefix> (prm) : GBL< IP128BGBL/prefix>
```

JSON command display format

The values described in **Display format contents** below are displayed in the following format as current IPv6 address set values.

```
{ "ip" : { "IP128BLLA" : <string>,  " IP128BGBL/prefix" : <string> , "prm_IP128BGBL/prefix" : <string> }
}
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
IP128BLLA	string	<IPv6 address>	To display the IPv6 link-local address.
IP128BGBL/prefix	string	IP128BGBL : <IPv6 address >	To display the IPv6 global address. The initial value is not set. If it is not set or assigned, "NONE" will be displayed.
		prefix : Prefix length	To display the prefix length of IPv6 global address. The initial value is 64.
prm_IP128BGBL/ prefix	string	prm_IP128BGBL : <IPv6 address >	To display the IPv6 global address of parameter area. The initial value is not set. If it is not set, "NONE" will be displayed.
		prefix : Prefix length	To display the prefix length of IPv6 global address in parameter area. The initial value is 64.

Example of command line usage

```
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>ip
ip LLA<fe80::21d:129f:35c5:9a>, GBL<NONE/64> (prm):GBL<NONE/64>
>
>ip 2001:db8::a
ip LLA<fe80::21d:129f:35c5:9a>, GBL<2001:db8::a/64> (prm):GBL<2001:db8::a/64>
>
>save
save parameter is saved
>
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>ip
ip LLA<fe80::21d:129f:35c5:9a>, GBL<2001:db8::a/64> (prm):GBL<2001:db8::a/64>
>
>init 1
init 1(BORDER)
inf 2b,62,0,5      {   FMng: changed fan join state (0 -> 5)     }
>
>ip 2001:db8::b
ip LLA<fe80::21d:129f:35c5:9a>, GBL<2001:db8::a/64> (prm):GBL<2001:db8::b/64>
>
>svrst
svrst parameter is saved and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>ip
ip LLA<fe80::21d:129f:35c5:9a>, GBL<2001:db8::b/64> (prm):GBL<2001:db8::b/64>
>
>init 2
init 2(ROUTER)
inf 2b,62,0,1      {   FMng: changed fan join state (0 -> 1)     }
>
>ip
ip LLA<fe80::21d:129f:35c5:9a>, GBL<NONE/64> (prm):GBL<2001:db8::b/64>
>
```

Example of JSON command usage

```
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "ip" : null }
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "NONE/64", "prm_IP128BGBL/prefix
" : "NONE/64" } }
>
>{ "ip" : "2001:db8::a"}
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "2001:db8::a/64", "prm_IP128BGBL
/prefix" : "2001:db8::a/64" } }
>
>{ "save" : null }
{ "save" : { "desc" : "parameter is saved" } }
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "ip" : null }
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "2001:db8::a/64", "prm_IP128BGBL
/prefix" : "2001:db8::a/64" } }
>
>{ "init" : "BORDER" }
{ "init" : "1(BORDER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 5 } }
>
>{ "ip" : "2001:db8::b" }
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "2001:db8::a/64", "prm_IP128BGBL
/prefix" : "2001:db8::b/64" } }
>
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{ "ip" : null }
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "2001:db8::b/64", "prm_IP128BGBL
/prefix" : "2001:db8::b/64" } }
>
>{ "init" : "ROUTER" }
{ "init" : "2(ROUTER)" }
>
>{ "ip" : null }
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "NONE/64", "prm_IP128BGBL/prefix
" : "2001:db8::b/64" } }
```

(12) init

Overview

This command starts the terminal in specified role (terminal operating mode).
It displays the current role (terminal operating mode).

Command-line format

```
init <role>
```

JSON command format

```
{ "init" : null }
{ "init" : <string> }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
role	string	"0" or "NONE"		Non-operation mode
		"1" or "BORDER"		Border Router terminal operation mode
		"2" or "ROUTER"	✓	Router terminal operation mode
		"3" or "LEAF"		Leaf terminal operation mode

If the argument is omitted, the current startup mode setting is displayed.
If the value of command name is set to null in JSON command, the current startup mode setting is displayed.

The change of terminal operation mode by this command is enabled only for the change from non-operation mode.
Changing from an operating mode to a non-operating mode and changing from an operating mode to another operating mode are disabled.

Command-line display format

The current role (terminal operation mode) is displayed in the following format with values described in **Display format contents** below.

```
init <role_num(role_name) >
```

JSON command display format

The role (terminal operation mode) is displayed in the following format with values described in **Display format contents** below.

```
{ "init" : <role_num(role_name) > }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
<role_num(role_name)>	string	role_num :0~3	To display the role number corresponding to the role (terminal operation mode) by “role_name” below. 0 : Non-operation mode 1 : Border Router terminal operation mode 2 : Router terminal operation mode 3 : Leaf terminal operation mode
		role_name : "NONE" "BORDER" "ROUTER" "LEAF"	To display the role (terminal operation mode) name corresponding to the above <role_num> number. "NONE" : Non-operation mode "BORDER": Border Router terminal operation mode "ROUTER" : Router terminal operation mode "LEAF" : Leaf terminal operation mode

Example of command line usage

<To set the role of Border Router>

```
>init
init 0(NONE)
>
>init 1
init 1(BORDER)
inf 2b,62,0,5       {   FMng: changed fan join state (0 -> 5)     }
…
inf 3c,39,70c,734   {    RPL: added node <2001:db8::2> => <2001:db8::1>}
>
```

<To set the role of Router>

```
>init 2
init 2(ROUTER)
inf 2b,62,0,1       {   FMng: changed fan join state (0 -> 1)     }
…
inf 2b,62,4,5       {   FMng: changed fan join state (4 -> 5)     }
>
```

<To set the role of Leaf>

```
>init LEAF
init 3(LEAF)
inf 2b,62,0,1       {   FMng: changed fan join state (0 -> 1)     }
…
inf 2b,62,4,5       {   FMng: changed fan join state (4 -> 5)     }
>
```

Example of JSON command usage

<To set the role of Border Router>

```
>{ "init" : null }
{ "init" : "0(NONE)" }
>
>{ "init" : "1" }
{ "init" : "1(BORDER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 5 } }
…
{ "inf" : { "cp" : 60, "cpnm" : "RPL", "en" : 57, "desc" : "added node", "IP128B" : "2001:db8::2", "PAREN
T_IP128B" : "2001:db8::1" } }
>
```

<To set the role of Router>

```
>{ "init" : "2" }
{ "init" : "2(ROUTER)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 1 } }
…
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 4, "now" :
 5 } }
>
```

<To set the role of Leaf>

```
>{ "init" : { "role" : "LEAF" } }
{ "init" : "3(LEAF)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 1 } }
…
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 4, "now" :
 5 } }
>
```

(13) atstart

Overview

This command sets the role (terminal operation mode) that is automatically set after restarting.
It displays the current role (terminal operation mode) setting value after reboot.

After setting the role (terminal operation mode) after restarting with this command, it will automatically start with the role (terminal operation mode) set by this command by saving the parameters with “3.3.8(2) save” and “3.3.8(4) svrst” commands.

Command-line format

```
atstart <role>
```

JSON command format

```
{ "atstart" : null }
{ "atstart" : <string> }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
role	string	"0" or "NONE"		Non-operation mode
		"1" or "BORDER"		Border Router terminal operation mode
		"2" or "ROUTER"	✓	Router terminal operation mode
		"3" or "LEAF"		Leaf terminal operation mode

If the argument is omitted, the current role (terminal operation mode) set value after restart is displayed.
If the value of command name is set to null in JSON command, the role (terminal operation mode) set value after the current restart will be displayed.

Command-line display format

The value described in **Display format contents** below is displayed in the following format as current role (terminal operation mode) set value after restart.

```
atstart <role_num(role_name) >
```

JSON command display format

The value described in **Display format contents** below is displayed in the following format as current role (terminal operation mode) set value after restart.

```
{ "atstart" : <role_num(role_name) > }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
<role_num(role_name) >	string	role_num : 0~3	To display the role number corresponding to role (terminal operation mode) by “role_name” below. 0 : Non-operation mode 1 : Border Router terminal operation mode 2 : Router terminal operation mode 3 : Leaf terminal operation mode
		role_name : "NONE" "BORDER" "ROUTER" "LEAF"	To display the role (terminal operation mode) name corresponding to the above <role_num> number. "NONE" : Non-operation mode "BORDER": Border Router terminal operation mode "ROUTER" : Router terminal operation mode "LEAF" : Leaf terminal operation mode

Example of command line usage

```
>atstart
atstart 0(NONE)
>
>atstart 1
atstart 1(BORDER)
>
>atstart BORDER
atstart 1(BORDER)
>
>atstart 2
atstart 2(ROUTER)
>
>atstart ROUTER
atstart 2(ROUTER)
>
>atstart 3
atstart 3(LEAF)
>
>atstart LEAF
atstart 3(LEAF)
>
```

Example of JSON command usage

```
>>{ "atstart" : null }
{ "atstart" : "3(LEAF)" }
>
>{ "atstart" : "1" }
{ "atstart" : "1(BORDER)" }
>
>{ "atstart" : "BORDER" }
{ "atstart" : "1(BORDER)" }
>
>{ "atstart" : { "role" : "1" } }
{ "atstart" : "1(BORDER)" }
>
>{ "atstart" : { "role" : "BORDER" } }
{ "atstart" : "1(BORDER)" }
>
>{ "atstart" : "2" }
{ "atstart" : "2(ROUTER)" }
>
>{ "atstart" : "ROUTER" }
{ "atstart" : "2(ROUTER)" }
>
>{ "atstart" : { "role" : "2" } }
{ "atstart" : "2(ROUTER)" }
>
>{ "atstart" : { "role" : "ROUTER" } }
{ "atstart" : "2(ROUTER)" }
>
>{ "atstart" : "3" }
{ "atstart" : "3(LEAF)" }
>
>{ "atstart" : "LEAF" }
{ "atstart" : "3(LEAF)" }
>
>{ "atstart" : { "role" : "3" } }
{ "atstart" : "3(LEAF)" }
>
>{ "atstart" : { "role" : "LEAF" } }
{ "atstart" : "3(LEAF)" }
>
```

(14) tcpcon

Overview

This command establishes the connection of TCP-Connection with the specified IPv6 address and port number (TCP-Server address / TCP-Server port). (Manual connection command)

"TCP-Server address" is the address of device terminal where the connection is established from the TCP port connection listening status.

Make a connection request to the TCP-Server port, which is in the Listen state on TCP-Server address device, and enter the address of device terminal to establish the connection as "TCP-Client address".

"TCP-Server port" is the port number on which a connection is established from the connection listening state on TCP-Server address device as the "TCP-Server port". Furthermore, the port number for establishing a connection is the "TCP-Server port" on TCP-Client address device terminal when a connection request is made from the TCP-Client address device terminal to TCP-Server address device terminal.

If TCP-Connection is established by this command, TCP data can be transmitted by “3.3.6(3) tcps” command.

Otherwise, if TCP-Connection has not been established before sending data with “3.3.6(3) tcps” command, the connection for establishing TCP-Connection is automatically established by on setting (automatic connection enabled) of "auto_connect" of “3.3.7(2) tcpopts” command.

This command does not accept command input until the connection processing for establishing TCP-Connection is completed.

Command-line format

```
tcpcon <ADDR> [port]
```

JSON command format

```
{ "tcpcon" : <string> }
{ "tcpcon" : { "ADDR" : <string>, "port" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	IPv6 unicast address connection
port	number	1~65534	Connection (TCP-Server) port number If this parameter is omitted, the default transmission TCP-Server port number will be used to establish TCP-Connection. The default transmission TCP-Server port number is the port number set in the "send_port" option of “3.3.7(2) tcpopts” command to establish a connection of TCP-Connection. (The default port number is 3610 = Echonet-Lite.)

Command-line display format

The connection results are displayed as follows.

Results	Description
Normal	If it ends normally, there will be no output display.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. tcpcon: invalid address.
Abnormal If connection failed	If the TCP-Connection cannot be established with IPv6 unicast address to which the argument ADDR is connected, the following string is displayed. tcpcon: failed to connect the TCP port.

JSON command display format

The connection results are displayed as follows.

Results	Description
Normal	If it ends normally, there will be no output display.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. { "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -12, "desc" : "invalid address" } }
Abnormal If connection failed	If the TCP-Connection cannot be established with IPv6 unicast address to which the argument ADDR is connected, the following character string is displayed. { "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -70, "desc" : "failed to connect the TCP port" } }

Example of command line usage

```
>tcpstat
tcpstat num:2
tcpstat CLOSED      <NONE>             0       <NONE>          3610
tcpstat LISTEN      <NONE>             44340   <NONE>          3610
>
>tcpcon 2001:db8::1 3610
>
>tcpstat
tcpstat num:2
tcpstat FIN_WAIT_1  <2001:db8::1>       3610    <2001:db8::2>   49153
tcpstat LISTEN      <NONE>             44340   <NONE>          3610
>
```

Example of JSON command usage

```
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "CLOSED", "REMOTE_ADDR" : "NONE", "remote_port" : 0, "LOCAL_ADDR" : "NONE",
 "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44340, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
>{ "tcpcon" : { "ADDR" : "2001:db8::1", "port" : 3610 } }
>
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::1", "remote_port" : 3610, "LOCAL_
ADDR" : "2001:db8::2", "local_port" : 49153 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44340, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

(15) tcpdis

Overview

This command disconnects TCP-Connection established to the specified IPv6 address and port number (TCP-Server address / TCP-Server port). (Manual disconnect command)
(For "TCP-Server address" / "TCP-Server port", Refer to the overview of the “3.3.2(14) tcpcon” command.)
When the TCP-Connection is disconnected by this command, TCP-Connection is automatically established again and data is transmitted based on the ON setting (automatic) of “3.3.7(2) tcpopts” command "auto_connect" (automatic connection enabled) during the transmission of TCP data on “3.3.6(3) tcps” command.

This command does not accept command input until disconnection process of the established TCP-Connection is completed.

Command-line format

```
tcpdis <ADDR> [port]
```

JSON command format

```
{ "tcpdis" : <string> }
{ "tcpdis" : { "ADDR" : <string>, "port" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	IPv6 unicast address connection. (disconnection target) (* Multicast address cannot be set)
port	number	1~65534	Connection port number. (TCP-Server to disconnect) If this parameter is omitted, the default transmission TCP-Server port number will be disconnected when TCP-Connection is established. The default transmission TCP-Server port number is the port number set in the "send_port" option of “3.3.7(2) tcpopts” command, and disconnects when TCP-Connection is established. (The default port number is 3610 = Echonet-Lite.)

Command-line display format

The connection result is displayed with the following contents.

Result	Description
Normal	If it ends normally, there will be no output display.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. tcpcon: invalid address.
Abnormal Not connected when IPv6 address and port number are specified	If it ends normally, there will be no output display. It does not output anything.

JSON command display format

The connection result is displayed with the following contents.

Result	Description
Normal	If it ends normally, there will be no output display. .
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -12, "desc" : "invalid address" } }`
Abnormal Not connected when IPv6 address and port number are specified	If it ends normally, there will be no output display.

Example of command line usage

```
>cpstat num:3
tcpstat LISTEN      <NONE>             44340   <NONE>          3610
tcpstat FIN_WAIT_1  <2001:db8::2>       49153   <2001:db8::1>   3610
tcpstat FIN_WAIT_1  <2001:db8::3>       49153   <2001:db8::1>   3610
>
>tcpdis 2001:db8::2 3610
>
>tcpstat
tcpstat num:2
tcpstat LISTEN      <NONE>             44340   <NONE>          3610
tcpstat FIN_WAIT_1  <2001:db8::3>       49153   <2001:db8::1>   3610
>
```

Example of JSON command usage

```
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 3 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::3", "remote_port" : 49153, "LOCAL
_ADDR" : "2001:db8::1", "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44340, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::2", "remote_port" : 49153, "LOCAL
_ADDR" : "2001:db8::1", "local_port" : 3610 } }
>
>{ "tcpdis" : { "ADDR" : "2001:db8::2", "port" : 3610 } }
>
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::3", "remote_port" : 49153, "LOCAL
_ADDR" : "2001:db8::1", "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44340, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

(16) rantsw

Overview

This command sets the antenna used by PHY (RADIO).
It displays the current setting of antenna used by PHY (RADIO).
The antenna settings that can be set depend on the platform.

Platform	Configurable Antenna Number	Initial Value	Supports Antenna Diversity
BP35C4(ROHM)	The rantsw command on the BP35C4 Platform is not implemented and cannot be used. (The antenna number is fixed at 1. It isnot compatible with antenna diversity.)		
BP35C5(ROHM)	1, 2	1	✓

Command-line format

```
rantsw [num]
```

JSON command format

```
{ "rantsw" : null }
{ "rantsw" : <number> }
{ "rantsw" : { "num" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
num	number	0	To work with antenna diversity.
		Wi-SUN module antenna number	To work in fixed antenna. The number of antennas that can be set depends on the Wi-SUN module. If an antenna number that does not exist is entered in Wi-SUN module, it will be set to diversity setting (0).

If the argument is omitted, the current antenna settings will be displayed below.
If the value of the command name is set to null in JSON command, the current antenna setting will be displayed.

Command-line display format

Similar to the above command-line format, the current setting value of the antenna used by PHY (RADIO) is displayed.

JSON command display format

Similar to the above JSON command format, the current setting value of the antenna used by PHY (RADIO) is displayed in JSON format.

Example of command line usage

```
>rantsw
rantsw 1
>rantsw 0
rantsw 0
>rantsw
rantsw 0
>rantsw 1
rantsw 1
>
```

Example of JSON command usage

```
>{ "rantsw" : null }
{ "rantsw" : 1 }
>{ "rantsw" : 0 }
{ "ranrsw" : 0 }
>{ "rantsw" : null }
{ "rantsw" : 0 }
>{ "rantsw" : { "num" : 1 } }
{ "rantsw" : 1 }
>
```

(17) rfec

Overview

This command sets the FEC (Forward Error Correction) function operation of PHY (RADIO).
It displays the current settings for FEC (Forward Error Correction) function operation of PHY (RADIO).

The FEC function operation of this command and PHY (RADIO) depends on the platform as follows.

Platform	Implementation status for each platform
BP35C4(ROHM)	- The rfec command on the BP35C4 platform has not been implemented, and the FEC function operation cannot be used.
BP35C5(ROHM)	✓ FEC function operation on the rfec command can be used. (*1)

*1 : The FEC function is an optional item of IEEE 802.15.4g.
It shall be based on the forward error correction (FEC) function.

Command-line format

```
rfec [enable]
```

JSON command format

```
{ "rfec" : null }
{ "rfec" : <bool> }
{ "rfec" : { "enable" : <bool> } }
```

Command argument

Argument Name	JSON Argument Type	Command-line Set Value	JSON Command Set Value	Initial Value	Description
enable	bool	0/off	0/false	✓	Disable FEC function
		Numbers other than 0 / on	Numbers other than 0 /true		Enable FEC function

If the connection status of FAN is not "INIT" (connection status number 0: initial status) (communication enabled status during connection or connection with other node terminals), the setting to enable (FEC function enable / disable switching operation) above is not sent to PHY (RADIO), but the setting is changed to the parameter area.
After setting this command, execute the “3.3.8(2) save” and “3.3.8(4) svrst”commands to save the parameters, and after restarting, it will automatically start with the set value of this command. (Refer to “3.3.4(4) fstat”for the FAN connection status.)
If the argument is omitted, the value of the current FEC function operation at the transmission rate of the PHY is displayed.
If the value of the command name is set to null in JSON command, the value of the current FEC function operation is displayed.

Command-line display format

The values described in the “**Display format contents**” below are displayed in the following format as set values related to current FEC function operation.

```
rfec <enable> (prm):<prm_enable>
```

JSON command display format

The values described in the “**Display format contents**” below are displayed in the following format as set values related to FEC function operation of the current PHY (RADIO).

```
{ "rfec" : { "enable" : <bool>, "prm_enable" : <bool> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
enable	bool	On or Off True or false in JSON command	To display the enable / disable setting for FEC function operation of the current PHY (RADIO).
prm_enable			To display the enable / disable setting for FEC operation in parameter area. (The set value after restart is displayed.)

Example of command line usage

```
>fstat
fstat 5(OPERATIONAL)
>
>rfec
rfec off (prm):off
>
>rfec on
rfec off (prm):on
>
>svrst
svrst parameter is saved and reset delay 0sec
inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 ROHM BP35C5(ROHM ML7436N:ML7421)
// Wi-SUN Profile for FAN (Nov 19 2020 12:07:07)
////////////////////////////////////////////////////////////
auto start 3 (LEAF)...
init 3(LEAF)
inf 2b,62,0,1      {   FMng: changed fan join state (0 -> 1)     }
>
>rfec
rfec on (prm):on
>
```

Example of JSON command usage

```
>{ "fstat" : null }
{ "fstat" : { "num" : 5, "desc" : "OPERATIONAL" } }
>
>{ "rfec" : null }
{ "rfec" : { "enable" : false, "prm_enable" : false } }
>
>{ "rfec" : true }
{ "rfec" : { "enable" : false, "prm_enable" : true } }
>
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 ROHM BP35C5(ROHM ML7436N:ML7421)", "ProfileBuild" : "Wi-SUN Profile for FAN (Nov 19 2020 12:07:07)
" } }
{ "init" : "3(LEAF)" }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 0, "now" :
 1 } }
>
>{ "rfec" : null }
{ "rfec" : { "enable" : true, "prm_enable" : true } }
>
```

(18) sleep

Overview

This command transitions the protocol stack and platform into standby mode (SUSPEND).
It restarts on the UART input.
Standby mode (SUSPEND) transition and return operations are platform-dependent.

Platform	Implementation Status for each Platform
BP35C4(ROHM)	- The sleep command on the BP35C4 platform has not been implemented and cannot be used.
BP35C5(ROHM)	✓ It can be used in the Leaf terminal operation mode and the FAN connection state. (* 1 to * 5)

*1 : This command can only be used when role (terminal operation mode) of Leaf is started in terminal operation mode by the "3.3.2(12) init" and "3.3.2(13) atstart"commands.
*2 : The transition to standby mode (SUSPEND) on this command can shift only when FAN connection status on "3.3.4(4) fstat" is 5 ("OPERATIONAL").
*3 : Automatic restart from standby mode (SUSPEND) is performed by the key input from UART (BP35C5-UART0).
*4 : Restart from *3 starts the FAN connection status = 3 ("WAIT-PANCFG" status) according to "3.3.4(4) fstat", and automatically restarts up to FAN connection status = 5 ("OPERATIONAL") status.
*5 : While the restart is being executed in *4 (until the FAN connection status becomes 5 ("OPERATIONAL")), each command of the data transmission on 3.3.6 Data transmission commands"may cause an error.

Command-line format

```
sleep
```

JSON command format

```
{ "sleep" : null }
```

Command argument / Command-line display format / JSON command display format

Nothing (The argument of this command is not required. There is no display output content for executing this command. There will be no output.)

Example of command line usage

```
>fstat
fstat 5(OPERATIONAL)
>
>sleep
inf 4c,38,0,0      {    RPL: disconnected from DODAG(lct=0)      }
inf 40,2c,59e0,0   {   NBRS: deleted neighbor <001d129f35c500fd> }
(Shift to standby(SUSPEND) mode)
(Return by key input)
inf 2b,62,5,3      {   FMng: changed fan join state (5 -> 3)     }
inf 40,2b,59e0,2edd {   NBRS: added new neighbor <001d129f35c500fd>}
inf 2b,62,3,4      {   FMng: changed fan join state (3 -> 4)     }
inf 4c,3c,9dc0,1d00 {    RPL: preferred parent <001d129f35c500fd> }
inf 40,2d,9d70,59e0 {   NBRS: added address <2000::1> to <001d129f35c500fd>}
inf 4c,37,0,0      {    RPL: connected to DODAG                 }
inf 2b,62,4,5      {   FMng: changed fan join state (4 -> 5)     }
>(Return is complete)
```

Example of JSON command usage

```
>{ "sleep" : null }
{ "inf" : { "cp" : 64, "cpnm" : "NBRS", "en" : 44, "desc" : "deleted neighbor", "MAC64B" : "001d129f35c50
0fd" } }
{ "inf" : { "cp" : 76, "cpnm" : "RPL", "en" : 56, "desc" : "disconnected from DODAG" } }
(Shift to standby(SUSPEND) mode)
(Return by key input)
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 5, "now" :
 3 } }
{ "inf" : { "cp" : 64, "cpnm" : "NBRS", "en" : 43, "desc" : "added new neighbor", "MAC64B" : "001d129f35c
500fd" } }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 3, "now" :
 4 } }
{ "inf" : { "cp" : 76, "cpnm" : "RPL", "en" : 60, "desc" : "preferred parent", "MAC64B" : "001d129f35c500
fd" } }
{ "inf" : { "cp" : 64, "cpnm" : "NBRS", "en" : 45, "desc" : "added address", "IP128B" : "2000::1", "MAC64
B" : "001d129f35c500fd" } }
{ "inf" : { "cp" : 76, "cpnm" : "RPL", "en" : 55, "desc" : "connected to DODAG" } }
{ "inf" : { "cp" : 43, "cpnm" : "FMng", "en" : 98, "desc" : "changed fan join state", "prev" : 4, "now" :
 5 } }
>(Return is complete)
```

3.3.3 Border Router control connection setting display command

(1) leaseip

Overview

This command sets the fixed IPv6 global address issued by DHCPv6 server to MAC address.
It displays the registration information of fixed IPv6 global address issued by current DHCPv6 server.

The number of MAC addresses of nodes that can be registered depends on the platform.

Platform	Configurable Number
BP35C4(ROHM)	100
BP35C5(ROHM)	100

The following settings can be made in the fixed IPv6 global address settings issued by DHCPv6 server.
- Register a fixed IPv6 global address issued by DHCPv6 server for MAC address.
- Delete the registered fixed IPv6 global address.
- Delete all registered fixed IPv6 global addresses.

Registration information will not be displayed when this command is set.

Command-line format

```
leaseip <cmd> <MAC64B> <IP128BGBL>
```

JSON command format

```
{ "leaseip" : null }
{ "leaseip" : { "cmd" : <string>,"MAC64B" : <string>, "IP128BGBL" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
cmd	string	"add"	MAC address of argument 2 <MAC64B> EUI64 format is assigned to IPv6 address of argument 3 <IP128BGBL>. (Argument 2 <MAC64B> and argument 3 <IP128BGBL> are both required.)
		"del"	The address specified by argument 2 <MAC64B> or argument 3 <IP128BGBL> is deleted from the fixed IPv6 global address setting registered in "add" above. Either argument 2 <MAC64B> or argument 3 <IP128BGBL> is omitted.
		"clr"	To delete all settings issued to IPv6 address. It is not necessary to input argument 2 <MAC64B> and argument 3 <IP128BGBL>.
MAC64B	string	<MAC address>	To specify the EUI64 format MAC address that performs the operation set in Argument 1 <cmd>.
IP128BGBL	string	<IPv6 address>	To specify the IPv6 global address that performs the operation set in Argument 1<cmd>.

The IPv6 global addresses that can be set are as follows.

IPv6 Unique Local Address (ULA) (Address block = fc00::/7)
IPv6 Global Unicast Address (GUA) (Address block = 2000::/3)

If the argument is omitted, the registration information of IPv6 global address issued by DHCPv6 server is displayed.
If the value of the command name is set to null in JSON command, the registered information of IPv6 global address issued by DHCPv6 server will be displayed.

Command-line display format

The value described in the following "**Display format contents**" is displayed in the following format for the registered information of IPv6 address issued by DHCPv6 server.

```
leaseip num:<num>, max:<max>
leaseip <MAC64B>,  <IP128BGBL>
…(After that, the number of registered units is displayed.)
```

JSON command display format

The value described in the following "**Display format contents**" is displayed in the following format for the registered information of IPv6 address issued by DHCPv6 server.

```
{ "leaseip" : { "num" : <number>, "max" : <number> } }
{ "leaseip" : { "MAC64B" : <string>, "IP128BGBL" : <string> } }
…(After that, the number of registered units is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	More than 0	To display the current setting of the number of registered node terminals.
max	number	Platform-dependent	To display the current operation of the maximum number of units that can be registered on the platform.
MAC64B	string	<EUI64 MAC address>	To display the EUI64 format MAC address associated with the fixed assignment of IPv6 global address.
IP128BGBL	string	<IPv6 address>	To display the IPv6 address issued to <MAC64B> above.

Example of command line usage

```
>leaseip
leaseip num:0, max:16
>
>leaseip add 0011223344556670 2001:db8::10
>
>leaseip add 0011223344556671 2001:db8::11
>
>leaseip add 0011223344556672 2001:db8::12
>
>leaseip add 0011223344556673 2001:db8::13
>
>leaseip add 0011223344556674 2001:db8::14
>
>leaseip
leaseip num:5, max:16
leaseip <0011223344556670>, <2001:db8::10>
leaseip <0011223344556671>, <2001:db8::11>
leaseip <0011223344556672>, <2001:db8::12>
leaseip <0011223344556673>, <2001:db8::13>
leaseip <0011223344556674>, <2001:db8::14>
>
>leaseip del 0011223344556671
>
>leaseip del 2001:db8::12
>
>leaseip del 0011223344556674 2001:db8::14
>
>leaseip
leaseip <0011223344556670>, <2001:db8::10>
leaseip <0011223344556673>, <2001:db8::13>
>
>leaseip clr
>
>leaseip
leaseip num:0, max:16
>
```

Example of JSON command usage

```
>{ "leaseip" : null }
{ "leaseip" : { "num" : 0, "max" : 16 } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B" : "0011223344556670", "IP128BGBL" : "2001:db8::10" } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B" : "0011223344556671","IP128BGBL" : "2001:db8::11" } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B " : "0011223344556672","IP128BGBL" : "2001:db8::12" } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B" : "0011223344556673", "IP128BGBL" : "2001:db8::13" } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B" : "0011223344556673", "IP128BGBL" : "2001:db8::13" } }
>
>{ "leaseip" : { "cmd": "add", "MAC64B" : "0011223344556674", "IP128BGBL" : "2001:db8::14" } }
>
>{ "leaseip" : null }
{ "leaseip" : { "num" : 5, "max" : 16 } }
{ "leaseip" : { "MAC64B" : "0011223344556670", "IP128BGBL" : "2001:db8::10" } }
{ "leaseip" : { "MAC64B" : "0011223344556671", "IP128BGBL" : "2001:db8::11" } }
{ "leaseip" : { "MAC64B" : "0011223344556672", "IP128BGBL" : "2001:db8::12" } }
{ "leaseip" : { "MAC64B" : "0011223344556673", "IP128BGBL" : "2001:db8::13" } }
{ "leaseip" : { "MAC64B" : "0011223344556674", "IP128BGBL" : "2001:db8::14" } }
>
>{ "leaseip" : { "cmd" : "del", " MAC64B" : "0011223344556671" } }
>
>{ "leaseip" : { "cmd" : "del", " IP128BGBL" : "2001:db8::12" } }
>
>{ "leaseip" : { "cmd" : "del", " MAC64B" : "0011223344556674", "IP128BGBL" : "2001:db8::14" } }
>
>{ "leaseip" : null }
{ "leaseip" : { "num" : 2, "max" : 16 } }
{ "leaseip" : { "MAC64B" : "0011223344556670", "IP128BGBL" : "2001:db8::10" } }
{ "leaseip" : { "MAC64B" : "0011223344556673", "IP128BGBL" : "2001:db8::13" } }
>
>{ "leaseip" : "clr" }
>
>{ "leaseip" : null }
{ "leaseip" : { "num" : 0, "max" : 16 } }
>
```

(2) leaserng

Overview

This command sets the range of IPv6 address issued by DHCPv6 server.
It displays the range of IPv6 address issued by current DHCPv6 server.

Command-line format

```
leaserng <range>
```

JSON command format

```
{ "leaserng" : null }
{ "leaserng" : <number> }
{ "leaserng" : { "range" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial value	Description
range	number	1~65535	1000	To specify the number of IPv6 address allocation ranges by DHCPv6 server in decimal. The IPv6 address to be issued has the following range. IPv6 global address of DHCPv6 server (Border Router) + 1~ IPv6 global address of DHCPv6 server (Border Router) + argument <range>.

Set the IPv6 address range issued by DHCPv6 server so that it falls within the following address range.

IPv6 Unique Local Address (ULA) (Address block = fc00::/7)
IPv6 Global Unicast Address (GUA) (Address block = 2000::/3)

When the number of IPv6 addresses issued to DHCPv6 client exceeds the range above, DHCPv6 server will respond that it cannot be issued.
(In addition, within the range of the above IP address allocation, the fixed hand out IPv6 address associated with the MAC address by the "3.3.3(1) leaseip" command will be given priority.)
If the argument is omitted, the range of the issued IPv6 address of the current DHCPv6 server is displayed.
If the value of the command name is set to null in JSON command, the range of the IPv6 address issued by the current DHCPv6 server is displayed.

Command-line display format

The values described in **Display format contents** below is displayed in the following format, for the range of current DHCPv6 server issued IPv6 address.

```
leaserng <minIP128B> <-> <maxIP128B> (<range>)
```

JSON command display format

The value described in **Display format contents** below is displayed in the following format for the registered information of IPv6 address issued by DHCPv6 server.

```
{ "leaserng" : { "range" : <number>, "minIP128B" : <string>, "maxIP128B" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
range	number	0 or more	To display the range from the minimum value to maximum value of the above DHCP issued IPv6 address.
minIP128B	string	<IPv6 address>	To display the minimum value of the DHCP issued IPv6 address. (Border Router IPv6 global address + 1)
maxIP128B	string	<IPv6 address>	To display the maximum value of the DHCP issued IPv6 address.

In the following state, <minIP128B> <maxIP128B> is not displayed and 0 address "<::>" is displayed in IPv6 notation.
1: When init 2 (ROUTER) / init 3 (LEAF) is running
2: When IPv6 global address is not set

Example of command line usage

```
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>leaserng
leaserng <::> <-> <::> (1000)
>
>leaserng 65535
leaserng <::> <-> <::> (65535)
>
>init 1
init 1(BORDER)
>
>ip
ip LLA<fe80::21d:129f:35c5:9a>, GBL<2001:db8::1/64> (prm):GBL<NONE/64>
>
>leaserng
leaserng <2001:db8::2> <-> <2001:db8::1:0> (65535)
>
>reset
reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>init 2
init 2(ROUTER)
>
>leaserng
leaserng <::> <-> <::> (1000)
>
```

Example of JSON command usage

```
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{"leaserng" : null}
{ "leaserng" : { "range" : 1000, "minIP128B" : "::", "maxIP128B" : "::" } }
>
>{"leaserng" : 65535}
{ "leaserng" : { "range" : 65535, "minIP128B" : "::", "maxIP128B" : "::" } }
>
>{ "init" : "BORDER" }
{ "init" : "1(BORDER)" }
>
>{"ip" : null}
{ "ip" : { "IP128BLLA" : "fe80::21d:129f:35c5:9a", "IP128BGBL/prefix" : "2001:db8::1/64", "prm_IP128BGBL
/prefix" : "NONE/64" } }
>
>{"leaserng" : null}
{ "leaserng" : { "range" : 65535, "minIP128B" : "2001:db8::2", "maxIP128B" : "2001:db8::1:0" } }
>
>{ "reset" : null }
{ "reset" : 0 }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Aug 28 2020 18:07:04)" } }
{ "init" : "0(NONE)" }
>
>{"init" : "ROUTER"}
{ "init" : "2(ROUTER)" }
>
>{"leaserng" : null}
{ "leaserng" : { "range" : 1000, "minIP128B" : "::", "maxIP128B" : "::" } }
>
```

(3) nodef

Overview

On the Border Router, set the filter settings for the nodes that can be authenticated and connected by the MAC address.
It displays the filter settings of the nodes that can be currently authenticated and connected.
The number of MAC addresses of the nodes that can be registered depends on the platform.

Platform	Configurable Number
BP35C4(ROHM)	100
BP35C5(ROHM)	100

Command-line format

```
nodef [role] [MAC64B]
```

JSON command format

```
{ "nodef" : null }
{ "nodef" : <string> }
{ "nodef" : { "role" : <string> } }
{ "nodef" : { "role" : <string>, "MAC64B" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
role	string	"allow"	✓	Allow settings. To allow the authentication connection of the MAC address specified by the argument [MAC64B] without filtering. If the argument [MAC64B] is omitted, the default rule is changed to allow connection.
		"deny"		Deny setting. To filter the authentication connection of the MAC address specified by the argument [MAC64B] and denies the connection. If the argument [MAC64B] is omitted, the default rule is changed to deny connection.
		"del"		Rule deletion. To delete the rule setting for the MAC address specified by the argument [MAC64B]. Authentication connections from deleted MAC address terminals are filtered according to the default rules.
		"clr"		To delete all rules. Deletes all the currently set MAC address terminal rule settings. The default rules do not change. Authentication connections from deleted MAC address terminals are filtered according to the default rules.
MAC64B	string	<EUI64MAC address>	-	To specify EUI64 format MAC address to operate with the argument [role].

If the argument is omitted, the current rule settings will be displayed below.
If the value of the command name is set to null in JSON command, the current rule setting will be displayed below.

Command-line display format

The value described in "**Display format contents**" below is used to display the received packet filtering settings in the current MAC layer in the following format.

```
nodef num:<num>, max:<max>
nodef default  ( <default>  )
nodef <MAC64B> ( <role> )
nodef <MAC64B> ( <role> )
…(After that, the number of registered units is displayed.)
```

JSON command display format

The value described in "**Display format contents**" below is used to display the received packet filtering settings in current MAC layer in the following format.

```
{ "nodef" : { "num" : <number>, "max" : <number> } }
{ "nodef" : { "default" : <string> } }
{ "nodef" : { "role" : <string>, "MAC64B" : <string> } }
{ "nodef" : { "role" : <string>, "MAC64B" : <string> } }
…(After that, the number of registered units is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0 or more	To display the current setting of the registered node terminals.
max	number	Platform-dependent	To display the current operation maximum number of units that can be registered on the platform.
default	string	"allow"	Allow settings. To allow authenticated connections without filtering. The <role> of <MAC64B> below allows connections from all terminals other than "deny" (= denied setting) without filtering authentication connections.
		"deny"	Deny setting. To filter and deny authenticated connections. The <role> of <MAC64B> below filters authentication connections from all terminals other than "allow" (= permission is set) and rejects the connection.
MAC64B	string	<EUI64MAC address>	The terminal MAC address applied with the following <role>.
role	string	"allow"	Allow settings. To allow authentication connections from the above <MAC64B> address terminal without filtering.
		"deny"	Deny setting. To filter and reject authenticated connections from the above <MAC64B> address terminal.

Example of command line usage

<To allow all connections from MAC addresses not set in the rule>

```
>nodef allow
```

<To deny all connections from MAC addresses not set in the rule>

```
>nodef deny
```

<To deny a connection from the 0011223344556677>

```
>nodef deny 0011223344556677
```

<To allow a connection from the 0011223344556677>

```
>nodef allow 0011223344556688
```

<To show the current rule settings>

```
>nodef
nodef num:2, max:16
nodef default                 ( deny  )
nodef <0011223344556677>       ( deny  )
nodef <0011223344556688>       ( allow )
```

<To delete the rule of 0011223344556677>

```
>nodef del 0011223344556677
```

<To delete all registered rules>

```
>nodef clr
```

Example of JSON command usage

<To allow all connections from MAC addresses not set in the rule>

```
>{ "nodef" : "allow" }
```

<To deny all connections from MAC addresses not set in the rule>

```
>{ "nodef" : { "role" : "deny"} }
```

<To deny a connection from the 0011223344556677>

```
>{ "nodef" : { "role" : "deny",  "MAC64B" : "0011223344556677" } }
```

<To allow a connection from the 0011223344556677>

```
>{ "nodef" : { "role" : "allow", "MAC64B" : "0011223344556688" } }
```

<To show the current rule settings>

```
>{ "nodef" : null }
{ "nodef" : { "num" : 2, "max" : 16 } }
{ "nodef" : { "default" : "deny" } }
{ "nodef" : { "role" : "deny", "MAC64B" : "0011223344556677" } }
{ "nodef" : { "role" : "allow", "MAC64B" : "0011223344556688" } }
```

<To delete the rule of 0011223344556677>

```
>{ "nodef" : { "role" : "del",  "MAC64B" : "0011223344556677" } }
```

<To delete all registered rules>

```
>{ "nodef" : "clr" }
```

(4) fnode

Overview

This command displays the list of MAC addresses and IPv6 addresses of authenticated / connected nodes on the Border Router.
The authenticated / connected node information can be deleted. When deleting node information, make sure that node is turned off or out of range. If the power is turned off after deleting the node information, authentication / connection may be performed again before the node is turned off.
The number of nodes that are authenticated and connected depends on the platform.

Platform	Number of units that can be registered
BP35C4(ROHM)	100
BP35C5(ROHM)	100

Command-line format

```
fnode [cmd] [MAC64B]
fnode [cmd] [IP128B]
```

JSON command format

```
{ "fnode" : null }
{ "fnode" : { "cmd" : <string> } }
{ "fnode" : { "cmd" : <string>, "MAC64B" : <string> } }
{ "fnode" : { "cmd" : <string>, "IP128B" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
cmd	string	"del"	To delete authenticated / connected node information of the address (IPv6 address) specified by argument [MAC64B] or [IP128B].
		"clr"	To delete all authenticated / connected node information inside the stack.
MAC64B	string	<EUI64MAC address>	To specify EUI64 format MAC address to delete the authenticated / connected node information for "del" in the argument cmd above.
IP128B	string	<IPv6 address>	To specify IPv6 address to delete the authenticated / connected node information for "del" in the argument cmd above.

If the argument is omitted, the current authenticated / connected node is displayed.
If the value of the command name is set to null in JSON command, the existing authenticated / connected nodes will be displayed below.

Command-line display format

The value described in "**Display format contents**" below is used to display the current authenticated / connected node in the following format.

```
fnode num:<num>  max:<max>
fnode <MAC64B>,   <IP128B>
fnode <MAC64B>,   <IP128B>
…(After that, the number of authenticated / connected nodes is displayed.)
```

JSON command display format

The value described in "**Display format contents**" below is used to display the current authenticated / connected node in the following format.

```
{ "fnode" : { "num" : <num>, "max" : <num> } }
{ "fnode " : { "MAC64B" : <string>, "IP128B" : <string> } }
{ "fnode " : { "MAC64B" : <string>, "IP128B" : <string> } }
…(After that, the number of authenticated / connected nodes is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0 or more	To display the current number of authenticated / connected node terminals.
max	number	Platform-dependent	To display the current operation of maximum connected devices on the platform.
MAC64B	string	<EUI64MAC address>	The terminal MAC address of the authenticated / connected node.
IP128B	string	<IPv6 address>	The terminal IPv6 address of the authenticated / connected node.

Exammple of command line usage

<To show the current authenticated/connected node>

```
>fnode
fnode num:2 max:16
fnode <0011223344556677>,   <2001:db8::2>
fnode <0011223344556688>,   <2001:db8::3>
```

<To delete the authentication/connection information of 0011223344556677>

```
>fnode del 0011223344556677
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>
```

<To delete the authentication/connection information of 2001:db8::2>

```
>fnode del 2001:db8::2
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>
```

<To delete the information of all authenticated/connected nodes>

```
>fnode clr
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>         }
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::3>         }
```

Example of JSON command usage

<To show the current authenticated/connected node>

```
>{"fnode" : null}
{ "fnode" : { "num" : 2, "max" : 16 } }
{ "fnode" : { "MAC64B" : "0011223344556677", "IP128B" : "2001:db8::2" } }
{ "fnode" : { "MAC64B" : "0011223344556688", "IP128B" : "2001:db8::3" } }
```

<To delete the authentication/connection information of 0011223344556677>

```
>{"fnode":{"cmd":"del","MAC64B":"0011223344556677"}}
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>        }
```

<To delete the authentication/connection information of 2001:db8::2>

```
>{"fnode":{"cmd":"del","IP128B":"2001:db8::2"}}
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>        }
```

<To delete the information of all authenticated/connected nodes>

```
>{ "fnode" : { "cmd" : "clr" } }
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::2>         }
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::3>         }
```

3.3.4 Connection status display commands

(1) stat

Overview

This command displays the terminal information.

Command-line format

```
stat
```

JSON command format

```
{ "stat" : null }
```

Command argument

None

Command-line display format

The terminal information is displayed in the following format with values described in the "**Display format contents**".

```
stat <asoc>,<role>,<chan>,<MAC64B>,<PAN16B>,<IP128BLLA>
```

JSON command display format

The terminal information is displayed in the following format with values described in the "**Display format contents**".

```
{ "stat" : { "asoc" : <number>, "role" : <number>, "chan" : <number>, "MAC64B" : <string>, "PAN16B" : <st
ring>, "IP128BLLA" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
asoc	number	0 or 1	To display the PAN participation status (Link establishment status). 0: PAN not participating state (non-Link established state: communication with other nodes is not possible) 1: PAN participation status (Link establishment status: communication enabled status with other nodes)
role	number	0~3	To display the number corresponding to the following role (terminal operation mode). 0: Non-operation mode 1: Border Router terminal operation mode 2: Router terminal operation mode 3: Leaf terminal operation mode
chan	number	0~128	To display the current PHY configured channel number.
MAC64B	string	<EUI64MAC address>	To display the MAC address in EUI64 format.
PAN16B	string	<PAN-ID>	To display the PAN ID.
IP128BLLA	string	< IPv6 link local address>	To display the IPv6 link-local address generated by MAC address.

Example of command line usage

```
>stat
stat 1,1,33,<0001020304050601>,<cafe>,<fe80::201:203:405:601>
>
```

Example of JSON command usage

```
>{ "stat" : null }
{ "stat" : { "asoc" : 1, "role" : 1, "chan" : 33, "MAC64B" : "0001020304050601", "PAN16B" : "cafe", "IP12
8BLLA" : "fe80::201:203:405:601" } }
>
```

(2) rstat

Overview

This command displays the physical layer (PHY-RADIO) statistics.

Command-line format

```
rstat
```

JSON command format

```
{ "rstat" : null }
```

Command argument

None

Command-line display format

The physical layer (PHY-RADIO) statistical information is displayed in the following format with values described in "**Display format contents**" below.

```
rstat TX <tx_num>pkt <tx_err>errs <tx_bytes>bytes
rstat RX <rx_num>pkt <rx_err>errs <rx_bytes>bytes
rstat Other errs:<errs>
```

JSON command display format

The physical layer (PHY-RADIO) statistical information is displayed in the following format with values described in "**Display format contents**" below.

```
{ "rstat" : { "tx_num" : <number>, "tx_err" : <number>, "tx_bytes" : <number>, "rx_num" : <number>, "rx_e
rr" : <number>, "rx_bytes" : <number>, "errs" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
tx_num	number	0 or more	To display the number of transmitted packets.
tx_err			To display the transmission error counter.
tx_bytes			To display the transmitted number of bytes.
rx_num			To display the number of transmitted packets.
rx_err			To display the transmission error counter.
rx_bytes			To display the transmitted number of bytes.
errs			To display the other number of error counters.

Example of command line usage

```
>rstat
rstat TX 0033pkt 0000errs 3683bytes
rstat RX 0031pkt 0000errs 3478bytes
rstat Other errs:0
>
```

Example of JSON command usage

```
>{ "rstat" : null }
{ "rstat" : { "tx_num" : 33, "tx_err" : 0, "tx_bytes" : 3683, "rx_num" : 31, "rx_err" : 0, "rx_bytes" : 3
478, "errs" : 0 } }
>
```

(3) mstat

Overview

This command displays the MAC layer statistics.

Command-line format

```
mstat
```

JSON command format

```
{ "mstat" : null }
```

Command argument

None

Command-line display format

The MAC statistics are displayed in the following format with values described in "**Display format contents**" below.

```
mstat uptime <upmm>min <upss>sec
mstat limit <tst>msec(max:<mst>) in <lspn>pkt(max:<lmpn> drop:<dpn> cca:<ccapn>) available <lavlb>bytes
mstat send total:<tspn> (ok:<tsspn> retry:<tsrpn> err:<tsepn>)
mstat recv total:<trpn> (ok:<trspn> err:<trepn>)
```

JSON command display format

The MAC statistics are displayed in the following format with values described in "**Display format contents**" below.

```
{ "mstat" : { "upmm" : <number>, "upss" : <number>, "tst" : <number>, "mst" : <number>, "lspn" : <number>
, "lmpn" : <number>, "dpn" : <number>, "ccapn" : <number>, "lavlb" : <number>, "tspn" : <number>, "tsspn"
 : <number>, "tsrpn" : <number>, "tsepn" : <number>, "trpn" : <number>, "trspn" : <number>, "trepn" : <nu
mber> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
upmm	number	0 or more	To display the elapsed time (minutes) since the terminal started.
upss			To display the elapsed time (seconds) since the terminal started.
tst			To display the current cumulative transmission time (msec) in total suppression unit (30 seconds).
mst			To display the past maximum cumulative transmission time (msec) in total suppression unit (30 seconds).
lspn			To display the current total number of transmitted packets in total suppression unit (30 seconds).
lmpn			To display the maximum number of cumulative transmitted packets in past total suppression unit (30 seconds).
dpn			To display the total number of discarded packets.
ccapn			To display the total number of CCA (busy) detections.
lavlb			To display the number of bytes that can be sent up to current transmission limit in total suppression unit (30 seconds).
tspn			To display the cumulative number of packets sent (trials).
tsspn			To display the cumulative number of successful packets sent.
tsrpn			To display the cumulative number of retry packets sent.
tsepn			To display the cumulative number of failed packets.
trpn			To display the cumulative number of received packets.
trspn			To display the cumulative number of successful packets received.
trepn			To display the cumulative number of failed packets. The conditions for determining reception failure are as follows. - Not addressed to yourself (except when MAC filtering is performed) - When the format of the received packet is incorrect (CRC error, etc.)

Example of command line usage

```
>mstat
mstat uptime 8min 28sec
mstat limit 0msec (max:5) in 2pkt (max:31 drop:0 cca:0) available 325byte
mstat send total:33 (ok:33 retry:0 err:0)
mstat recv total:36 (ok:36 err:0)
>
```

Example of JSON command usage

```
>{ "mstat" : null }
{ "mstat" : { "upmm" : 8, "upss" : 28, "tst" : 0, "mst" : 5, "lspn" : 2, "lmpn" : 31, "dpn" : 0, "ccapn" :
 0, "lavlb" : 325, "tspn" : 33, "tsspn" : 33, "tsrpn" : 0, "tsepn" : 0, "trpn" : 36, "trspn" : 36, "trepn
" : 0 } }
>
```

(4) fstat

Overview

This command displays the FAN connection status.

Command-line format

```
fstat
```

JSON command format

```
{ "fstat" : null }
```

Command argument

None

Command-line display format

The value described in "**Display format contents**" below is used to display FAN connection status in the following format.

```
fstat <num>(<desc>)
```

JSON command display format

The value described in "**Display format contents**" below is used to display the FAN connection status in the following format.

```
{ "fstat" : { "num" : <number>, "desc" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0~5	FAN connection status number. Details of connection status number and overview areas follows.
desc	string	"INIT"	FAN connection status number 0: To display the initial status.
		"SELECT-PAN"	FAN connection status number 1: To display the status when the PAN to connect is selected.
		"AUTHENTICATE"	FAN connection status number 2: To display the status during authentication.
		"WAIT-PANCFG"	FAN connection status number 3: To represent the frame wait state from upper Router encrypted by authentication.
		"ROUTING"	FAN connection status number 4: To represent the network layer initialization state, including the RPL routing configuration state.
		"OPERATIONAL"	The FAN connection status number above: 5 To display the communication operational status that can be connected to other node terminals.

When it started in Border Router terminal operation mode, the above status skips the status of 1 to 4 and changes from 0 to 5 according to the FAN regulations.

When it started in Router terminal operation mode or Leaf terminal mode, the steps are sequentially stepped up from 0 → 1 → 2 → 3 → 4 → 5 according to the connection status.

Example of command line usage

<Not connected>

```
>fstat
fstat 0(INIT)
```

<After connecting: After completing the RPL network connection>

```
>fstat
fstat 5(OPERATIONAL)
```

Example of JSON command usage

<Not connected>

```
>{ "fstat" : null }
{ "fstat" : { "num" : 0, "desc" : "INIT" } }
```

<After connecting: After completing the RPL network connection>

```
>{ "fstat" : null }
{ "fstat" : { "num" : 5, "desc" : "OPERATIONAL" } }
```

(5) chconfig

Overview

This command shows the current set value for PHY settings, including area / transmission rate of the frequency used for PHY.

Command-line format

```
chconfig
```

JSON command format

```
{ "chconfig" : null }
```

Command argument

None

Command-line display format

The value described in "**Display format contents**" below is displayed in the following format as the set value related to current PHY setting.

```
chconfig domain=<domain>,rate=<rate>Kbps,class=<class>,chbase=<chbase>MHz,chspace=<freq_space>KHz,chnum
=<chnum>
```

JSON command display format

The value described in "**Display format contents**" below is displayed in the following format as set value related to the PHY setting.

```
{ "chconfig" : { "domain" : <number>, "rate" : <number>, "class" : <number>, "chbase" : <number>, "freq_s
pace" : <number>, "chnum"  : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
domain	number	<Area number (country code)>	To display the frequency area (Regulatory Domain) number to use.
rate	number	50/100/150/300	To display the PHY transmission rate set value in Kbps, which is defined by frequency area (Regulatory Domain) to use and band class (Operating Class) of the frequency area.
class	number	0/1/2	To display the operating class number of frequency area defined by frequency area used and transmission rate used.
chbase	number	<Frequency>	To display the frequency of channel 0 in area, as defined by frequency area (Regulatory Domain) to be used and the band class (Operating Class) of frequency area. To display in MHz on the command line. The JSON command is displayed in KHz units.
freq_space	number	< Frequency >	To display frequency width of the channel used in the area, defined by the frequency area to be used (Regulatory Domain) and the band class of the frequency area (Operating Class) in KHz.
chnum	number	0~128	To display the total number of channels in the area, defined by frequency area (Regulatory Domain) to be used and the band class (Operating Class) of frequency area.

Each of the above set values shall be based on "Wi-SUN PHY Specification".

Example of command line usage

```
>chconfig
chconfig domain=2,rate=150Kbps,class=2,chbase=920.900MHz,chspace=400KHz,chnum=18
>
>chrate 50
chrate 50Kbps (prm):50Kbps
>
>chconfig
chconfig domain=2,rate=50Kbps,class=1,chbase=920.600MHz,chspace=200KHz,chnum=38
>
```

Example of JSON command usage

```
>{ "chconfig" : null }
{ "chconfig" : { "domain" : 2, "rate" : 150, "class" : 2, "chbase" : 920900, "chspace" : 400, "chnum" : 1
8 } }
>
>{ "chrate" : 50 }
{ "chrate" : { "rate" : 50, "prm_rate" : 50 } }
>
>{ "chconfig" : null }
{ "chconfig" : { "domain" : 2, "rate" : 50, "class" : 1, "chbase" : 920600, "chspace" : 200, "chnum" : 38
 } }
>
```

(6) chcur

Overview

This command displays the current PHY configuration channel number.

Command-line format

```
chcur
```

JSON command format

```
{ "chcur" : null }
```

Command argument

None

Command-line display format

The current PHY setting channel number is displayed in the following format with values described in "**Display format contents**" below.

```
chcur <num>
```

JSON command display format

The current PHY setting channel number is displayed in the following format with values described in "**Display format contents**" below.

```
{ " chcur" : <number> }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	33~ 59 or 61	To display the current PHY configuration channel number.

Example of command line usage

```
>chcur
chcur 45
>
```

Example of JSON command usage

```
>{ "chcur" : null }
{ "chcur" : 45 }
>
```

(7) fmseckey

Overview

This command displays the MAC security key generated by FAN authentication connection operation.

Command-line format

```
fmseckey
```

Command argument

None

Command-line display format

The four MAC security keys are displayed in the following format with values described in "**Display format contents**" below.

```
fmseckey [0] <key1>
fmseckey [1] <key2>
fmseckey [2] <key3>
fmseckey [3] <key4>
```

JSON command display format

The four MAC security keys are displayed in the following format with values described in "**Display format contents**" below.

```
{ "fmseckey" : { "key1" : <string>, "key2" : <string>, "key3" : <string>, "key4" : <string>, } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
key1	string	32-byte character string converted from binary 16 bytes to hexadecimal ASCII	To display the four MAC security key strings generated by the FFAN authentication connection operation with a 32-byte string converted from binary 16 bytes to hexadecimal ASCII.
key2	string		
key3	string		
key4	string		

Example of command line usage

<Connecting>

```
>fmseckey
fmseckey [0] 516fa3080ea4d9c1a1aefa26deac9272
fmseckey [1] 00000000000000000000000000000000
fmseckey [2] 00000000000000000000000000000000
fmseckey [3] 00000000000000000000000000000000
>
```

Example of JSON command usage

<Connecting>

```
>{ "fmseckey" : null }
{ "fmseckey" : { "key1" : "516fa3080ea4d9c1a1aefa26deac9272", "key2" : "00000000000000000000000000000000
", "key3" : "00000000000000000000000000000000", "key4" : "00000000000000000000000000000000" } }
>
```

(8) nebr

Overview

This command displays information on nearby terminals (terminals detected within the radio range of your terminal).
The maximum number of terminals that can be detected by neighboring devices depends on the platform.

Platform	Configurable Number
BP35C4(ROHM)	128
BP35C5(ROHM)	128

Command-line format

```
nebr
```

JSON command format

```
{ "nebr" : null }
```

Command argument

None

Command-line display format

The values described in "**Display format contents**" below are displayed in the following format as information on nearby terminals (terminals detected within the radio range of your terminal).

```
nebr num:<num>, max:<max>, rank:<rank>(<DAGRANK>)
nebr <MAC64B>:     <ETX>/<EWMA>(<rssi>)        <IP128BGBL>
(<MAC64B> / <ETX> / <EWMA> / <rssi> / <IP128BGBL> are displayed hereafter for the number of detected unit
s)
```

JSON command display format

The values described in "**Display format contents**" below are displayed in the following format as information on nearby terminals (terminals detected within the radio range of your terminal).

```
{ "nebr" : { "num" : <number>, "max" : <number>, "rank" : <number>, "DAGRANK" : <number> } }
{ "nebr" : { "MAC64B" : <string>, "ETX" : <number>, "EWMA" : <number>, "rssi" : <number>, "IP128BGBL" : <
string> } }
(<MAC64B> / <ETX> / <RSSI> <rssi> <IP128BGBL> are displayed hereafter for the number of detected units)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0 or more	To display the number of nearby terminals (the number of nearby terminals that can send and receive within radio detection range).
max	number	Platform-dependent	To display the current operation maximum number of devices that can be detected by neighboring terminals on the platform.
rank	number	0~65535	To display the RPL network local terminal RANK (decimal number display).
DAGRANK	number	0~65535	To display the RPL network local terminal DAGRANK (decimal number display).
MAC64B	string	<MAC address>	To display the MAC address of the detected neighboring terminal.
ETX	number	0~255	To display the radio wave strength ETX value (decimal number display) of the detected neighboring terminal.
EWMA	number	0~255	To display the radio wave strength EWMA value (exponential moving average) (decimal number display) of the detected neighboring terminal.
rssi	number	-128~127	To display the signal strength (decimal number display) received at the end of the detected neighboring terminal.
IP128BGBL	string	<IPv6 address>	To display IPv6 global address of the detected neighboring terminal.

Example of command line usage

<Disconnecting>

```
>nebr
nebr num:0, max:64, rank:65535(0)
>
```

<Connecting>

```
>nebr
nebr num:2, max:64, rank:312(2)
nebr <0001020304050601>: 184/139(-35)   <2001:db8::1>
nebr <0001020304050603>: 240/139(-35)   <2001:db8::3>
>
```

Example of JSON command usage

<Disconnecting>

```
>{ "nebr" : null }
{ "nebr" : { "num" : 0, "max" : 64, "rank" : 65535, "DAGRANK" : 0 } }
>
```

<Connecting>

```
>{ "nebr" : null }
{ "nebr" : { "num" : 2, "max" : 64, "rank" : 312, "DAGRANK" : 2 } }
{ "nebr" : { "MAC64B" : "0001020304050601", "ETX" : 184, "RSL" : 139, "rssi" : -35, "IP128BGBL" : "2001:d
b8::1" } }
{ "nebr" : { "MAC64B" : "0001020304050603", "ETX" : 240, "RSL" : 139, "rssi" : -35, "IP128BGBL" : "2001:d
b8::3" } }
>
```

(9) parent

Overview

This command displays the address information of the currently set parent node.

Command-line format

```
parent
```

JSON command format

```
{ "parent" : null }
```

Command argument

None

Command-line display format

The address information of the parent node currently set in the following format is displayed with value described in the following "**Display format contents**".

```
parent MAC<MAC64B>,LLA<IP128BLLA>,GBL<IP128BGBL>
```

JSON command display format

The address information of the parent node currently set in the following format is displayed with value described in the following "**Display format contents**".

```
{ "parent" : { "MAC64B" : <string>, "IP128BLLA" : <string>, "IP128BGBL" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
MAC64B	string	<MAC address>	To display MAC address of the currently set parent node.
IP128BLLA	string	<IPv6 address>	To display IPv6 link local address of the currently set parent node.
IP128BGBL	string	<IPv6 address>	To display IPv6 global address of the currently set parent node.

Example of command line usage

<Disconnecting>

```
>parent
parent MAC<NONE>,LLA<NONE>,GBL<NONE>
>
```

<Connecting>

```
>parent
parent MAC<0001020304050601>,LLA<fe80::201:203:405:601>,GBL<2001:db8::1>
>
```

Example of JSON command usage

<Disconnecting>

```
>{ "parent" : null }
{ "parent" : { "MAC64B" : "NONE", "IP128BLLA" : "NONE", "IP128BULA" : "NONE", "IP128BGBL" : "NONE" } }
>
```

<Connecting>

```
>{ "parent" : null }
{ "parent" : { "MAC64B" : "0001020304050601", "IP128BLLA" : "fe80::201:203:405:601", "IP128BGBL" : "2001
:db8::1" } }
>
```

(10) rplinf

Overview

This command displays information about the RPL.

Command-line format

```
rplinf
```

JSON command format

```
{ "rplinf" : null }
```

Command argument

None

Command-line display format

Information about RPL is displayed in the following format with values described in "**Display format contents**" below.

```
rplinf DODAG ID    : <DODAG>
rplinf instance ID : <instance>
rplinf my rank     : <rank>
rplinf my DAGRANK  : <DAGRANK>
rplinf parent      : MAC<MAC64B>,LLA<IP128BLLA>,GBL<IP128BGBL>
```

JSON command display format

Information about RPL is displayed in the following format with values described in "**Display format contents**" below.

```
{ "rplinf" : { "DODAG" : <string>, "instance" : <number>, "rank" : <number>, "DAGRANK" : <number>, "MAC64
B" : "<string>", "IP128BLLA" : "<string>", "IP128BGBL" : "<string>" } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
DODAG	string	<IPv6 address>	To display the currently set DODAG-ID address.
instance	number	0~255	To display the currently set DODAG instance ID.
rank	number	0~65535	To display the local terminal RANK of RPL network.
DAGRANK	number	0~65535	To display the local terminal DAGRANK of RPL network.
MAC64B	string	<MAC address>	To display MAC address of the currently set parent node.
IP128BLLA	string	<IPv6 address>	To display IPv6 link local address of the currently set parent node.
IP128BGBL	string	<IPv6 address>	To display IPv6 global address of the currently set parent node.

Example of command line usage

<Disconnecting>

```
>rplinf
>
```

<Connecting: Border Router>

```
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 128
rplinf my DAGRANK  : 1
rplinf parent      : MAC<NONE>,LLA<NONE>,GBL<NONE>
>
```

<Connecting: Router/Leaf>

```
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 276
rplinf my DAGRANK  : 2
rplinf parent      : MAC<0001020304050601>,LLA<fe80::201:203:405:601>,GBL<2001:db8::1>
>
```

Example of JSON command usage

<Disconnecting>

```
>{ "rplinf" : null }
>
```

<Connecting: Border Router>

```
>{ "rplinf" : null }
{ "rplinf" : { "DODAG" : "<2001:db8::1>", "instance" : 0, "rank" : 128, "DAGRANK" : 1, "MAC64B" : "NONE",
 "IP128BLLA" : "NONE", "IP128BGBL" : "NONE" } }
>
```

<Connecting: Router/Leaf>

```
>{ "rplinf" : null }
{ "rplinf" : { "DODAG" : "2001:db8::1", "instance" : 0, "rank" : 276, "DAGRANK" : 2, "MAC64B" : "00010203
04050601", "IP128BLLA" : "fe80::201:203:405:601", "IP128BGBL" : "2001:db8::1" } }
>
```

(11) tcpstat

Overview

This command displays the TCP-Connection status.

Command-line format

```
tcpstat
```

JSON command format

```
{ "tcpstat" : null }
```

Command argument

None

Command-line display format

Information about RPL is displayed in the following format with values described in "**Display format contents**" below.

```
tcpstat num:<num>
tcpstat <tcp_state>        <REMOTE_ADDR>      <remote_port>    <LOCAL_ADDR>   <local_port>
…(After a few minutes, TCP-Connection is displayed.)
```

JSON command display format

Information about RPL is displayed in the following format with values described in "**Display format contents**" below.

```
{ "tcpstat" : { "num" : <number> } }
{ "tcpstat" : { "tcp_state" : <string>, "REMOTE_ADDR" : <string>, "remote_port" : <number>, "LOCAL_ADDR"
 : <string>, "local_port" : <number> } }
…(After a few minutes, TCP-Connection is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	<IPv6 address>	To display the number of sessions currently being generated.
tcp_state	string	Character string on the right	To display the Connection status (TCP status). The Connection status (TCP status) displays the following. "CLOSED" "LISTEN" "SYN_SENT" "SYN_RCVD" "ESTABLISHED" "FIN_WAIT_1" "FIN_WAIT_2" "CLOSE_WAIT" "CLOSING" "LAST_ACK" "TIME_WAIT"
REMOTE_ADDR	string	<IPv6 address>	To display the (destination) remote address of TCP-Connection.
remote_port	number	<MAC address>	To display the (destination) remote port number of TCP-Connection.
LOCAL_ADDR	string	<IPv6 address>	To display the (source) local address of TCP-Connection.
local_port	number	<IPv6 address>	To display the (source) local port number of TCP-Connection.

Example of command line usage

<Disconnecting>

```
>tcpstat
tcpstat num:2
tcpstat CLOSED      <NONE>             0       <NONE>          3610
tcpstat LISTEN      <NONE>             44565   <NONE>          3610
>
```

<Connecting: Connected to the LISTEN port>

```
>tcpstat
tcpstat num:2
tcpstat FIN_WAIT_1  <2001:db8::2>      49153   <2001:db8::1>   3610
tcpstat LISTEN      <NONE>             44565   <NONE>          3610
>
```

<Connecting: Connected to the LISTEN port of the destination>

```
tcpcon 2001:db8::1
>
>tcpstat
tcpstat num:2
tcpstat FIN_WAIT_1  <2001:db8::1>      3610    <2001:db8::2>   49153
tcpstat LISTEN      <NONE>             44565   <NONE>          3610
>
```

Example of JSON command usage

<Disconnectting>

```
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "CLOSED", "REMOTE_ADDR" : "NONE", "remote_port" : 0, "LOCAL_ADDR" : "NONE",
 "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44565, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

<Connecting: Connected to the LISTEN port>

```
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::2", "remote_port" : 49153, "LOCAL
_ADDR" : "2001:db8::1", "local_port" : 3610 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44565, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

<Connecting: Connected to the LISTEN port of the destination>

```
>{ "tcpcon" : { "ADDR" : "2001:db8::1", "port" : 3610 } }
>
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::1", "remote_port" : 3610, "LOCAL_
ADDR" : "2001:db8::2", "local_port" : 49153 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 44565, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

3.3.5 Border Router control connection status display commands

(1) rplsr

Overview

This command displays the RPL source routing table maintained by Border Router. (Only when Border Router is operating)

Command-line format

```
rplsr
```

JSON command format

```
{ "rplsr" : null }
```

Command argument

None

Command-line display format

The address information of the parent node currently set in the following format is displayed with value described in the following "**Display format contents**".

```
rplsr - Routing links (<num> in total):
rplsr -- <IP128B> to <IP128B_PARENT> (lifetime: <lifetime> seconds)
…(After that, the number of connected devices is displayed.)
```

JSON command display format

The address information of the parent node currently set in the following format is displayed with value described in the following "**Display format contents**".

```
{ "rplsr" : { "num" : <number> } }
{ "rplsr" : { "IP128B" : <string>, "IP128B_PARENT" : <string>, "lifetime" : <number> } }
…(After that, the number of connected devices is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	string	0 or more	Number of registrations (number of links) in the RPL connection source routing table
IP128B	string	<IPv6 address>	IPv6 address of the terminal connected to Border by RPL
IP128B_PARENT	string	<IPv6 address>	IPv6 address of the parent (adjacent Router) terminal connected by RPL to Border
lifetime	number	0 or more	RPL validity time (The RPL connection will be updated until this becomes 0. When it becomes 0, the registration information will be deleted.)

Example of command line usage

<Routing information is not registered>

```
>rplsr
- Routing links (0 in total):
>
```

<Routing information is registered>

```
>rplsr
- Routing links (2 in total):
-- 2001:db8::2 to 2001:db8::1 (lifetime: 7154 seconds)
-- 2001:db8::3 to 2001:db8::2 (lifetime: 7156 seconds)
>
```

Example of JSON command usage

<Routing information is not registered>

```
>{ "rplsr" : null }
{ "rplsr" : { "num" : 0 } }
>
```

<Routing information is registered>

```
>{ "rplsr" : null }
{ "rplsr" : { "num" : 2 } }
{ "rplsr" : { "IP128B" : "2001:db8::2", "IP128B_PARENT" : "2001:db8::1", "lifetime" : 7196 } }
{ "rplsr" : { "IP128B" : "2001:db8::3", "IP128B_PARENT" : "2001:db8::2", "lifetime" : 7144 } }
>
```

(2) leased

Overview

This command shows the list of IPv6 global addresses currently issued by DHCPv6 server. (Only when Border Router is operating)
The maximum number of addresses that can be assigned by DHCPv6 server depends on the platform.

Platform	Configurable Number
BP35C4(ROHM)	128
BP35C5(ROHM)	128

Command-line format

```
leased
```

JSON command format

```
{ "leased" : null }
```

Command-line display format

The value described in the "**Display format contents**" below is displayed in the following format, and the information of DHCPv6 server currently issued is displayed.

```
leased num:<num>, max:<max>
leased <IP128BGBL>,         <MAC64B>
(Afterwards, the issued IPv6 global address is displayed.)
```

JSON command display format

The value described in the "**Display format contents**" below is displayed in the following format, and the information of the DHCPv6 server currently issued is displayed.

```
{ "leased" : { "num" : <number>, "max" : <number> } }
{ "leased" : { "IP128BGBL" : <string>, "MAC64B" : <string> } }
(Afterwards, the issued IPv6 global address is displayed.)
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
num	number	0 or more	To display the current number of issued node terminals.
max	number	Platform-dependent	To display current operation of the maximum number of addresses that can be issued on platform.
IP128BGBL	string	<IPv6 address>	To display the currently issued IPv6 global address.
MAC64B	string	<MAC address>	To display MAC address for the IPv6 global address issued above.

Example of command line usage

<Connecting>

```
>leased
leased num:2, max:256
leased <2001:db8::2>,       <0001020304050602>
leased <2001:db8::3>,       <0001020304050603>
>
```

Example of JSON command usage

<Connecting>

```
>{ "leased" : null }
{ "leased" : { "num" : 2, "max" : 256 } }
{ "leased" : { "IP128BGBL" : "2001:db8::2", "MAC64B" : "0001020304050602" } }
{ "leased" : { "IP128BGBL" : "2001:db8::3", "MAC64B" : "0001020304050603" } }
>
```

3.3.6 Data transmission commands

(1) udps

Overview

This command transmits data using UDP protocol.

Command-line format

```
udps <ADDR> [port] <data>
```

JSON command format

```
{ "udps" : { "ADDR" : <string>, "data" : <string> } }
{ "udps" : { "ADDR" : <string>, "port" : <number>, "data" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	Destination IPv6 address
port	number	1~65534	Outgoing port number If this parameter is omitted, data will be transmitted using the default transmission port number. The default transmission port number is 3610 (Echonet-Lite), but if you set the port number in the "send_port" option of the “3.3.7(1) udpopts” command, data will be sent using the specified port number.
data	string	Hexadecimal ASCII string	UDP packet transmission data payload Enter the data payload as a hexadecimal ASCII string. Two characters make one byte of data. The data payload is up to 1024 Octets (input data is 2048 Bytes in ASCII).

Command-line display format

The connection result is displayed with the following contents.

Result	Description
Normal	If it ends normally, there is no output content display. It does not output anything.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. udps: invalid address.
Abnormal data is invalid	If the argument data is invalid, the following character string is outputted. udps: invalid data.

JSON command display format

The connection result is displayed with the following contents.

Result	Description
Normal	If successful, there will be no return of command.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -12, "desc" : "invalid address" } }`
Abnormal data is invalid	If the argument data is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -4, "desc" : "invalid data." } }`

Refer to “4.3.1(1) udpr” for the display when receiving UDP.
Refer to “3.3.7(1) udpopts” and “4.3.2(1) udpsd” for the display of completed transmission by udps command.

Example of command line usage

<To send to unicast address >

```
>udps 2001:db8::2 00112233445566
```

<To send to Link-Local multicast address>

```
>udps ff02::1 00112233445566
```

<To send to Realm-Local multicast address>

```
>udps ff03::1 00112233445566
```

<Specifying the sending port number>

```
>udps 2001:db8::3 5000 00112233445566
```

<Specifying the default sending port with udpopts and send on the port>

```
>udpopts disp_port on
>udpopts send_done on
>udpopts send_port 5001
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
udpopts send_port=5001
udpopts listen_port=[0]5001
>
>udps 2001:db8::2 0011223344556677
udpsd <2001:db8::2> (5001)
>
```

Example of JSON command usage

<To send to unicast address>

```
>{ "udps" : { "ADDR" : "2001:db8::2", "data" : "00112233445566" } }
```

<To send to Link-Local multicast address>

```
>{ "udps" : { "ADDR" : "ff02::1", "data" : "00112233445566" } }
```

<To send to Realm-Local multicast address>

```
>{ "udps" : { "ADDR" : "ff03::1", "data" : "00112233445566" } }
```

<Specifying the sending port number>

```
>{ "udps" : { "ADDR" : "2001:db8::3", "port" : 5001, "data" : "00112233445566" } }
```

<Specifying the default sending port with udpopts and send on the port>

```
>{ "udpopts" : { "option" : "disp_port", "enable" : true } }
>{ "udpopts" : { "option" : "send_done", "enable" : true } }
>{ "udpopts" : { "option" : "send_port", "port" : 5001 } }
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : true, "disp_rss
i" : false, "send_done" : true, "send_port" : 5001, "listen_port1" : 5001, "listen_port2" : 65535, "liste
n_port3" : 65535, "listen_port4" : 65535 } }
>
>{ "udps" : { "ADDR" : "2001:db8::3", "port" : 5001, "data" : "00112233445566" } }
{ "udpsd" : { "ADDR" : "2001:db8::3", "port" : 5001, "len" : 8 } }
>
```

(2) udpst

Overview

This command transmits the text string data using UDP protocol.

Command-line format

```
udpst <ADDR> [port] "<data>"
```

JSON command format

```
{ "udpst" : { "ADDR" : <string>, "data" : <string> } }
{ "udpst" : { "ADDR" : <string>, "port" : <number>, "data" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	Destination IPv6 address
port	number	1~65534	Outgoing port number If this parameter is omitted, data will be transmitted using the default transmission port number for UDP text transmission / reception. The default transmission port number is 20171, but if you set the port number with "send_port_text" option of “3.3.7(1) udpopts” command, data will be sent with the specified port number.
data	string	ASCII character string	UDP packet transmission data payload If the string you entered is a string literal enclosed in double quotation marks (`'"'`), the double quotation marks (`'"'`) are not included in the payload of the UDP packet you send. For details, refer to “2.2.2 Command-line”. Payload is limited to 1024 Octets (input data is 1024 bytes in ASCII).

Command-line display format

The connection result is displayed with the following contents.

Result	Description
Normal	If it ends normally, there is no output content display. It does not output anything.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. udpst: invalid address.
Abnormal data is invalid	If the argument data is invalid, the following character string is outputted. udpst: invalid data.

JSON command display format

The connection result is displayed with the following contents.

Result	Description
Normal	If successful, there will be no return of command.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -12, "desc" : "invalid address" } }`
Abnormal data is invalid	If the argument data is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -4, "desc" : "invalid data." } }`

Refer to “4.3.1(2) udprt” for the display of text data when receiving UDP.
Refer to “3.3.7(1) udpopts” and “4.3.2(1) udpsd” for the display of completed transmission by udpst command.

Example of command line usage

<To send to unicast address>

```
>udpst 2001:db8::2 "abcdef"
```

<To send to Link-Local multicast address>

```
>udpst ff02::1 "abcdef"
```

<To send to Realm-Local multicast address>

```
>udpst ff03::1 "abcdef"
```

<Specifying the sending port number>

```
>udpst 2001:db8::3 5000 "abcdef"
```

<Specifying the default sending port with udpopts and send on the port>

```
>udpopts disp_port on
>udpopts send_done on
>udpopts send_port_text 5001
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
udpopts send_port=3610 send_port_text=5001
udpopts listen_port=3610
udpopts listen_port_text=20171
>
>udpst 2001:db8::2 "12345678"
udpsd <2001:db8::2> (5001)
>
```

<Send with space character (' ')>

```
>udpopts send_done on
>udpopts disp_len on
>udpst 2001:db8::2 "1234 5678"
udpsd <2001:db8::2> (5001) (9)
>
```

<Send with double quote ('\"')>

```
>udpopts send_done on
>udpopts disp_len on
>udpst 2001:db8::2 "1234\"5678"
udpsd <2001:db8::2> (9)
>
```

<Send with tab character ('\t') and CR newline character ('\n')>

```
>udpopts send_done on
>udpopts disp_len on
>udpst 2001:db8::2 "1234\t\n5678"
udpsd <2001:db8::2> (10)
>
```

<Send the string "\n">

```
>udpopts send_done on
>udpopts disp_len on
>udpst 2001:db8::2 "1234\\n5678"
udpsd <2001:db8::2> (10)
>
```

Example of JSON command usage

<To send to unicast address>

```
>{ "udpst" : { "ADDR" : "2001:db8::2", "data" : "abcdef" } }
```

<To send to Link-Local multicast address>

```
>{ "udpst" : { "ADDR" : "ff02::1", "data" : "abcdef" } }
```

<To send to Realm-Local multicast address>

```
>{ "udpst" : { "ADDR" : "ff03::1", "data" : "abcdef" } }
```

<Specifying the sending port number>

```
>{ "udpst" : { "ADDR" : "2001:db8::3", "port" : 5001, "data" : "abcdef" } }
```

<Specifying the default sending port with udpopts and send on the port>

```
>{ "udpopts" : { "option" :  "disp_port", "enable" : true } }
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "send_port_text", "port" : 5001 } }
{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -22, "desc" : "overflowed the buffer" } }
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : true, "disp_rss
i" : false, "send_done" : true, "send_port" : 3610, "send_port_text" : 5001, "listen_port1" : 3610, "list
en_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "listen
t_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
>{ "udpst" : { "ADDR" : "2001:db8::2", "port" : 5001,"data" : "abcdef" } }
{ "udpsd" : { "ADDR" : "2001:db8::2", "port" : 5001, "len" : 6 } }
>
```

<Send with space character (' ')>

```
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpst" : { "ADDR" : "2001:db8::2", "data" : "1234 5678" } }
{ "udpsd" : { "ADDR" : "2001:db8::2", "port" : 20171, "len" : 9 } }
>
```

<Send with double quote ('\"')>

```
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpst" : { "ADDR" : "2001:db8::2", "data" : "1234\"5678" } }
{ "udpsd" : { "ADDR" : "2001:db8::2", "port" : 20171, "len" : 9 } }
>
```

<Send with tab character ('\t') and CR newline character ('\n')>

```
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpst" : { "ADDR" : "2001:db8::2", "data" : "1234\t\n5678" } }
{ "udpsd" : { "ADDR" : "2001:db8::2", "port" : 20171, "len" : 10 } }
>
```

<Send the string "\n">

```
{ "udpopts" : { "option" :  "send_done", "enable" : true } }
{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
{ "udpst" : { "ADDR" : "2001:db8::2", "data" : "1234\\n5678" } }
>
```

(3) tcps

Overview

This command transmits data using the TCP protocol.
It sends TCP data to the specified IPv6 address and port number (TCP-Server address / TCP-Server port).
(For "TCP-Server address" / "TCP-Server port", refer to the overview of “3.3.2(14) tcpcon” command.)

If TCP-Connection is not established for the specified IPv6 address and port number (TCP-Server address / TCP-Server port), TCP-Connection is automatically established on the ON setting of "auto_connect" option of “3.3.7(2) tcpopts” command (automatic connection is enabled). Data transmission is performed after connecting the established TCP-Connection.

Refer to “4.3.1(3) tcpr” for the display during TCP data reception.
Refer to ”3.3.7(2) tcpopts” and “4.3.2(2) tcpsd” for the display of completed transmission by tcps command.

Command-line format

```
tcps <ADDR> [port] <data>
```

JSON command format

```
{ "tcps" : { "ADDR" : <string>, "data" : <string> } }
{ "tcps" : { "ADDR" : <string>, "port" : <number>, "data" : <string> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	IPv6 unicast address destination (* Multicast address cannot be set)
port	number	1~65534	Outgoing TCP-Server port number If this parameter is omitted, data will be transmitted using the default transmission TCP-Server port number. The default transmission TCP-Server port number is the port number set in "send_port" option of “3.3.7(2) tcpopts” command to establish a TTCP-Connection and transmit data. (The default port number is 3610 = Echonet-Lite.)
data	string	Hexadecimal ASCII string	TCP packet transmission data payload Enter the data payload as a hexadecimal ASCII string. Two characters make one byte of data. Payload is limited to 1024 Octets (input data is 2048 Bytes in ASCII).

Command-line display format

The connection result is displayed with the following contents.

Result	Description
Normal	If it ends normally, there is no output content display. It does not output anything.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. tcpcon: invalid address.
Abnormal If it cannot connect	If the TCP-Connection cannot be established to IPv6 unicast address to which the argument ADDR is connected, the following character string is outputted. tcps: failed to connect the TCP port.
Abnormal Data is invalid	If the argument data is invalid, the following character string is outputted. udps: invalid data.

JSON command display format

The connection result is displayed with the following contents.

Result	Description
Normal	If successful, there will be no return of command.
Abnormal ADDR is invalid	If the argument ADDR is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -12, "desc" : "invalid address" } }`
Abnormal If it cannot connect	If the TCP-Connection cannot be established to IPv6 unicast address to which the argument ADDR is connected, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -70, "desc" : "failed to connect the TCP port" } }`
Abnormal Data is invalid	If the argument data is invalid, the following character string is outputted. `{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -4, "desc" : "invalid data." } }`

Example of command line usage

<To send data with the default Send TCP-Server port number>

```
>tcps 2001:db8::1 0011223344556677
```

<Specifying the sending port number>

```
>tcps 2001:db8::1 5001 0011223344556677
```

<To send data using the specified default Send TCP-Server port number with the tcpopts command>

```
>tcpopts disp_port on
>tcpopts send_done on
>tcpopts send_port 5001
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=on send_done=on
tcpopts send_port=5001
tcpopts listen_port=[0]3610
tcpopts idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
>
>tcps 2001:db8::1 5001 0011223344556677
tcpsd <2001:db8::1> (5001)
>
```

Example of JSON command usage

<To send data with the default Send TCP-Server port number>

```
>{ "tcps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
```

<Specifying the sending port number>

```
>{ "tcps" : { "ADDR" : "2001:db8::1", "port" : 5000, "data" : "0011223344556677" } }
```

<To send data using the specified default Send TCP-Server port number with the tcpopts command>

```
>{ "tcpopts" : { "option" :  "disp_port", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_port", "port" : 5001 } }
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : true, "send_done"
 : true, "send_port" : 5001, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "list
en_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 10, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
>{ "tcps" : { "ADDR" : "2001:db8::1", "port" : 5000, "data" : "0011223344556677" } }
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 5000, "len" : 8 } }
>
```

(4) ping

Overview

This command transmits ping (ICMPv6 ECHO Request) to the specified address.

Command-line format

```
ping [cmd] <ADDR> [size] [sec] [count]
```

JSON command format

```
{ "ping" : { "cmd" : <string>, "ADDR" : <string>, "size" : <number>,  "sec" : <number>, "cnt" : <number>
} }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
cmd	string	start/stop/state	To specify the command of the following character string. "start": Starts regular ping transmission. Sends a ping packet at the cycle specified by argument interval. "stop": Stops ping transmission at regular intervals. "state": Displays the execution status of regular ping transmission. (If this argument is omitted, the following argument count won't be considered, and count = 1 single ping transmission operation will be performed.)
ADDR	string	<IPv6 address>	MAC address or IPv6 address destination
size	number	0~1024	To specify the payload length in bytes. (Default 32 Bytes)
sec	number	1 or more	Transmission interval (in seconds) during regular cycle execution (default 5 seconds)
count	number	0 or more	To specify the number of transmissions. (In the above argument cmd, specify the number of transmissions when "start" is specified. If this argument is omitted or 0 when "start" is specified in the argument cmd above, the number of transmissions is unlimited, and with the ping command, "stop" in the above argument cmd will send ping at regular intervals until the command is executed. If the above argument cmd is omitted, count = 1 single ping transmission operation will be performed.)

Command-line display format

When the ping transmission is completed, the transmission result will be displayed in the following format with values described in the "**Display format contents**" below.

```
<txcnt> transmitted, <rxcnt> received, <perlost>% loss (min=<min_sec>/max=<max_sec>/avr=<avr_sec> sec)
```

JSON command display format

When the ping transmission is completed, the transmission result will be displayed in the following format with values described in the "**Display format contents**" below.

```
{ "ping_state" : { "txcnt" : <number>, "rxcnt" : 1, "min_msec" : <number>, "max_msec" : <number>, "avr_ms
ec" : <number>, "total_msec" : <number> } }
```

Display format contents

(When ping transmission is completed)

Argument Name	JSON Argument Type	Display Value	Description
txcnt	number	0 or more	Number of completed transmission
rxcnt			ICMPv6 ECHO Reply reception count
perlost			Lost rate (displayed only on the command line)
min_sec			Minimum response time (sec) (displayed on command line only)
max_sec			Maximum response time (sec) (displayed on command line only)
avr_sec			Average response time (sec) (displayed on command line only)
min_msec			Minimum response time (msec) (displayed in JSON only)
max_msec			Maximum response time (msec) (displayed in JSON only)
avr_msec			Average response time (msec) (displayed in JSON only)
total_msec			Total response time (msec) (displayed in JSON only)

Refer to “4.3.1(4) ping (ping_recv)” for the display when receiving ICMPv6 ECHO Reply.

Example of command line usage

```
>ping 2001:db8::1
ping <2001:db8::1> (seq=1 sz=32bytes time=0.020sec) 1/1
1 transmitted, 1 received, 0.0% loss (min=0.020/max=0.020/avr=0.020 sec)
>
>ping 2001:db8::1 64
ping <2001:db8::1> (seq=1 sz=64bytes time=0.110sec) 1/1
1 transmitted, 1 received, 0.0% loss (min=0.110/max=0.110/avr=0.110 sec)
>
>ping start 2001:db8::1 64 2 64
ping <2001:db8::1> (seq=1 sz=64bytes time=0.100sec) 1/64
ping <2001:db8::1> (seq=2 sz=64bytes time=0.070sec) 2/64
ping <2001:db8::1> (seq=3 sz=64bytes time=0.070sec) 3/64
>ping state
3 transmitted, 3 received, 0.0% loss (min=0.070/max=0.100/avr=0.080 sec)
ping <2001:db8::1> (seq=4 sz=64bytes time=0.080sec) 4/64
ping <2001:db8::1> (seq=5 sz=64bytes time=0.050sec) 5/64
>ping stop
5 transmitted, 5 received, 0.0% loss (min=0.050/max=0.100/avr=0.074 sec)
>
```

Example of JSON command usage

```
>>{ "ping" : { "ADDR" : "2001:db8::1" } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 1, "size" : 32, "msec" : 100, "txcnt" : 1, "cnt" : 1, "
rank" : 330 } }
{ "ping_state" : { "txcnt" : 1, "rxcnt" : 1, "min_msec" : 100, "max_msec" : 100, "avr_msec" : 100, "total
_msec" : 100 } }
>
>{ "ping" : { "ADDR" : "2001:db8::1", "size" : 64 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 1, "size" : 64, "msec" : 90, "txcnt" : 1, "cnt" : 1, "r
ank" : 330 } }
{ "ping_state" : { "txcnt" : 1, "rxcnt" : 1, "min_msec" : 90, "max_msec" : 90, "avr_msec" : 90, "total_ms
ec" : 90 } }
>
>{ "ping" : { "cmd" : "start", "ADDR" : "2001:db8::1", "size" : 64, "sec" : 2, "cnt" : 64 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 1, "size" : 64, "msec" : 90, "txcnt" : 1, "cnt" : 64, "
rank" : 330 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 2, "size" : 64, "msec" : 50, "txcnt" : 2, "cnt" : 64, "
rank" : 330 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 3, "size" : 64, "msec" : 80, "txcnt" : 3, "cnt" : 64, "
rank" : 330 } }
>{ "ping" : { "cmd" : "state"} }
{ "ping_state" : { "txcnt" : 3, "rxcnt" : 3, "min_msec" : 50, "max_msec" : 90, "avr_msec" : 73, "total_ms
ec" : 220 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 4, "size" : 64, "msec" : 270, "txcnt" : 4, "cnt" : 64,
"rank" : 330 } }
{ "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 5, "size" : 64, "msec" : 20, "txcnt" : 5, "cnt" : 64, "
rank" : 330 } }
>{ "ping" : "stop" }
{ "ping_state" : { "txcnt" : 5, "rxcnt" : 5, "min_msec" : 20, "max_msec" : 270, "avr_msec" : 102, "total_
msec" : 510 } }
>
```

3.3.7 Data transmission / reception setting commands

(1) udpopts

Overview

This command displays the UDP data transmission / reception and UDP packet format setting and setting contents

The following settings and displays are carried out for UDP data transmission / reception.
(1) Sets the display of udpsd message to become the transmission completion display to the user by udps / udpst command and displays setting contents
(2) Sets the display of data length for udpr (data reception binary code output display), udprt (data reception text output display), and udpsd (data transmission completion display) and displays setting contents
(3) Sets the display of port number for udpr/ udprt/ udpsd messages and displays setting contents
(4) Sets the display of receiving signal strength for udpr / udprt messages and displays setting contents
(5) Sets the UDP-Listen port so that data can be received at specified UDP port for udpg messages and displays setting contents
(6) Sets the UDP-Listen port so that data can be received at specified UDP port for udprt messages and displays setting contents
(7) Sets the default transmission port for udps / udpst command and displays setting contents
The following settings and displays are carried out for UDP packet format.
(8) Setting of UDP Header Compression
(9) Setting of UDP CheckSum Elied

Command-line format

```
udpopts <option> <port|enable>
```

JSON command format

```
{ "udpopts" : null }
{ "udpopts" : { "option" : <stiring>, "enable" : <bool> } }
{ "udpopts" : { "option" : <stiring>, "port" : <number> } }
```

Command argument

Set parameters from argument “option” are set with argument “enable” or “port”

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
option	string	comp_hdr	enable=1	To set the UDP Header Compression. (For set value of this option, set with enable value below .)
		comp_csum	enable=0	To set the UDP CheckSum Elie. (For set value of this option, set with enable value below.)
		disp_len	enable=0	To set the transmission/reception data length display for udpr (received data binary code output display), udprt (received data text output display) and udpsd (transmission completion display). Always displayed in the udpr and udpsd display in JSON format. (For set value of this option, set with enable value below.)
		disp_port	enable=0	To set the transmission/reception port number display setting for reception display (udpr/udprt) / transmission completion display (udpsd) Always displayed in the udpr, udpsd and udpst display in JSON format. (For set value of this option, set with enable value below.)
		disp_rssi	enable=0	To set the Receiving signal strength (RSSI) display setting for reception display (udpr/ udprt). Always displayed in the udpr, udpsd and udpst display in JSON format. (For set value of this option, set with enable value below.)
		send_done	enable=0	To set the display setting for completion of transmission (udpsd) to user by udps and udpst command (For set value of this option, set with enable value below.)

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
		send_port	port=3610 (Echonet-Lite)	To set the default transmission port setting for udps command (*1, *2 (For set value of this option, set with port value below.)
		send_port _text	port=20171	To set the default transmission port setting for udps command (*1, *3 (For set value of this option, set with port value below.)
option	string	listen_port	port=3610 (Echonet-Lite)	To set Listen for receiving display of the specified UDP port in received data binary code display message (udpr) (*1) (For set value of this option, set with port value below.) The maximum number of ports that can be registered is 4 ports.
		listen_port _text	port=20171	To set Listen for receiving display of the specified UDP port in received data binary code display message (udprt) is set (*1) (For set value of this option, set with port value below.) The maximum number of ports that can be registered is 4 ports.
enable	bool	0(false) or 1(true)	-	To set enable/disable for send_done, disp_len, disp_port, disp_rssi, comp_hdr and comp_csum of the above arguments option
port	number	Port Number (1~65534) or (-1)	-	To set the port number for listen_port, listen_port_text, send_port and send_port_text of the above arguments option If listen_port and listen_port_text is specified in the argument option mentioned above and port number is set to -1, the added port number is deleted and the default sending port becomes the Initial Value of listen_port or listen_port_text.

*1: For the following UDP ports used in each protocol package in the stack, the message output of reception display (udpr / udprt)
and transmission completion display (udpsd) cannot be carried out even when Listen is set.
(UDP port number with invalid Listen setting)
・546(DHCPv6-DOWNSTREAM) ・547(DHCPv6-UPSTREM) ・716(PANA) ・1812(RADIUS-SERVER)
・9903(Multiple-PING) ・10253(EAPOL) ・19788(MLE) ・51812(RADIUS-CLIENT)
*2: If the transmission port setting is set to default for udps command by send_port, Listen will be automatically set to specified port.
*3: If the transmission port setting is set to default for udpst command by send_port_text, Listen will be automatically set to specified port.
*4: If send_port and send_port_text setting are performed in *2 and *3, the operation in automatic Listen setting will be less than the maximum number of ports that can be registered.

If the argument is omitted, the settings related to UDP data transmission / reception and UDP packet format will be displayed below.
If the value of the command name is null in JSON command, the display setting contents of the UDP data transmission / reception and the setting contents related to the UDP packet format will be displayed below.

Command-line format

Values described in "**Display format contents**" below are displayed in the following format as display setting for UDP data transmission / reception from current user and as setting related to the UDP packet format

```
udpopts comp_hdr=<comp_hdr> comp_csum=<comp_csum>
udpopts disp_len=<disp_len> disp_port=<disp_port> disp_rssi=<disp_rssi> send_done=<send_done>
udpopts send_port=<send_port>
udpopts send_port_text=<send_port_text>
udpopts listen_port=<listen_port1>,<listen_port2>,<listen_port3>,<listen_port4>
udpopts listen_port_text=<listen_port_text1>,<listen_port_text2>,<listen_port_text3>,<listen_port_text4
>
```

JSON command display format

Values described in "**Display format contents**" mentioned below are displayed in the following format as display setting for UDP data transmission / reception from user and as setting related to the UDP packet format

```
{ "udpopts" : { "comp_hdr" : <bool>, "comp_csum" : <bool>, "disp_len" : <bool>, "disp_port" : <bool>, "di
sp_rssi" : <bool>, "send_done" : <bool>, "send_port" : <number>, "send_port_text" : <number>, "listen_po
rt1" : <number>, "listen_port2" : <number>, "listen_port3" : <number>, "listen_port4" : <number>, "liste
nt_port_text1" : <number>, "listent_port_text2" : <number>, "listent_port_text3" : <number>, "listent_po
rt_text4" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
comp_hdr	bool	on or off true or false in JSON command	To display the contents of UDP Header Compression setting
comp_csum			To display the contents of UDP CheckSum Elied setting
disp_len			To display the setting for the display of transmission / reception data length in reception display (udpr/udprt) and transmission completion display (udpsd) messag
disp_port			To display the setting for the display of transmission/reception port number in reception display (udpr/udprt) and transmission completion display (udpsd) message
disp_rssi			To display the setting for display of receiving signal strength (RSSI) in receiving display (udpr / udprt) message
send_done			To display the setting of display for transmission completion (udpsd) message to user by udps command
send_port	number	port number (1~65534)	To display the default outbound port settings for udps command.
send_port_text			To display the default outbound port settings for udpst command.
listen_port1			To display the Listen ports 1 to 4 that enable reception of the specified UDP port in the Received data Binary code display message (udpr).
listen_port2			
listen_port3			
listen_port4			
listen_port_text1			To display the Listen ports 1 to 4 that enable reception of the specified UDP port in the received data text display message (udprt).
listen_port_text2			
listen_port_text3			
listen_port_text4			

Example of command line usage

<To show the setting value>

```
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=off disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
```

<To turn ON or OFF the display of transmission completion (send_done)>

```
>udpopts send_done off
>
>udps 2001:db8::1 0011223344556677
>
>udpst 2001:db8::1 "abcdef"
>>udpopts send_done on
>
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1>
>
>udpst 2001:db8::1 "abcdef"
udpsd <2001:db8::1>
>
```

<To turn ON or OFF the display of transmission/reception data length (disp_len)>

```
>udpopts send_done on
>udpopts disp_len off
>
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1>
>
>udpst 2001:db8::1 "abcdef"
udpsd <2001:db8::1>
>
>udpopts send_done on
>udpopts disp_len on
>
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1> (8)
>
>udpst 2001:db8::1 "abcdef"
udpsd <2001:db8::1> (6)
>
```

<To turn ON or OFF the display of transmission/reception port (disp_port)>

```
>udpopts send_done on
>udpopts disp_len  off
>udpopts disp_port off
>
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1>
>
>udpst 2001:db8::1 "abcdef"
udpsd <2001:db8::1>
>
>udpopts send_done on
>udpopts disp_len  off
>udpopts disp_port on
>
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1> (3610)
>
>udpst 2001:db8::1 "abcdef"
udpsd <2001:db8::1> (20171)
>
```

<To turn ON or OFF the display of received signal strength (disp_rss)>

```
>udpopts disp_len  off
>udpopts disp_port off
>udpopts disp_rssi off
>
udpr <2001:db8::1> 0011223344556677
udprt <2001:db8::1> "abcdef"
>
>udpopts disp_len  off
>udpopts disp_port off
>udpopts disp_rssi on
>
udpr <2001:db8::1> (-35) 0011223344556677
>
udprt <2001:db8::1> (-35) "abcdef"
>
```

<To set the Listen port (listen_port) so that the specified UDP port can be displayed in the received data binary code display message (udpr)>

```
>udpopts disp_len  off
>udpopts disp_rssi off
>udpopts disp_port on
>udpopts listen_port 5000
>
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610,5000
udpopts listen_port_text=20171
>
udpr <2001:db8::1> (5000) 0011223344556677
>
```

<To set the Listen port (listen_port_text) so that the specified UDP port can be displayed in the received data text display message (udprt)>

```
>udpopts disp_len  off
>udpopts disp_rssi off
>udpopts disp_port on
>udpopts listen_port_text 5001
>
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171,5001
>
udprt <2001:db8::1> (5001) "abcdef"
>
```

<To set the default send port (send_port) with the udps command>

```
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
udpopts send_port=5002 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
>udps 2001:db8::1 0011223344556677
>
udpsd <2001:db8::1> (5002)
>
```

<To set the default send port (send_port_text) with the udpst command>

```
>udpopts disp_port on
>udpopts send_done on
>udpopts send_port_text 5003
>
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
udpopts send_port=3610 send_port_text=5003
udpopts listen_port=3610
udpopts listen_port_text=20171
>
>udps 2001:db8::1 0011223344556677
>
udpsd <2001:db8::1> (3610)
>
```

<To turn ON or OFF the UDP header compression (comp_hdr)>

```
>udpopts comp_hdr on
>
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=off disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
>udpopts comp_hdr off
>
>udpopts
udpopts comp_hdr=off comp_csum=off
udpopts disp_len=off disp_port=off disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
```

<To turn ON or OFF the UDP checksum omission (comp_csum)>

```
>udpopts comp_csum on
>
>udpopts
udpopts comp_hdr=on comp_csum=on
udpopts disp_len=off disp_port=off disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
>udpopts comp_csum off
>
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=off disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610
udpopts listen_port_text=20171
>
```

Example of JSON command usage

<To show the setting value>

```
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "l
isten_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "lis
tent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
```

<To set the Listen port (listen_port) so that the specified UDP port can be displayed in the received data binary code display message (udpr)>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "listen_port", "port" : 5000 } }
>
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "l
isten_port2" : 5000, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "list
ent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
{ "udpr" : { "ADDR" : "2001:db8::1", "port" : 5000, "rssi" : -35, "len" : 8, "data" : "0011223344556677"
} }
>
```

<To set the default send port (send_port) with the udps command>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "send_port", "port" : 5002 } }
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : true, "send_port" : 5002, "send_port_text" : 20171, "listen_port1" : 3610, "li
sten_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "list
ent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
>{ "udps" : { "ADDR" : "2001:db8::1", "port" : 5002, "data" : "0011223344556677" } }
>
{ "udpsd" : { "ADDR" : "2001:db8::1", "port" : 5002, "len" : 8 } }
>
```

<To set the default send port (send_port_text) with the udpst command>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "send_done", "enable" : true } }
>{ "udpopts" : { "option" :  "send_port_text", "port" : 5003 } }
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : true, "send_port" : 3610, "send_port_text" : 5003, "listen_port1" : 3610, "lis
ten_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "liste
nt_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
>{ "udps" : { "ADDR" : "2001:db8::1", "port" : 5003, "data" : "0011223344556677" } }
{ "udpsd" : { "ADDR" : "2001:db8::1", "port" : 5003, "len" : 8 } }
>
```

<To turn ON or OFF the UDP header compression (comp_hdr)>

```
>{ "udpopts" : { "option" :  "comp_hdr", "enable" : true } }
>
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "l
isten_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "lis
tent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
>{ "udpopts" : { "option" :  "comp_hdr", "enable" : false } }
>
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : false, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_r
ssi" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "
listen_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "li
stent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
```

<To turn ON or OFF the UDP checksum omission (comp_csum)>

```
>{ "udpopts" : { "option" :  "comp_hdr", "enable" : true } }
>
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_rs
si" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "l
isten_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "lis
tent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
>{ "udpopts" : { "option" :  "comp_hdr", "enable" : false } }
>
>{ "udpopts" : null }
{ "udpopts" : { "comp_hdr" : false, "comp_csum" : false, "disp_len" : false, "disp_port" : false, "disp_r
ssi" : false, "send_done" : false, "send_port" : 3610, "send_port_text" : 20171, "listen_port1" : 3610, "
listen_port2" : 65535, "listen_port3" : 65535, "listen_port4" : 65535, "listent_port_text1" : 20171, "li
stent_port_text2" : 65535, "listent_port_text3" : 65535, "listent_port_text4" : 65535 } }
>
```

(2) tcpopts

Overview

This command displays the TCP data transmission / reception contents, and TCP packet setting and contents
For TCP data transmission / reception contents, the following settings and displays are carried out.
(1) Sets the display of TCP-Connection auto-establishment when sending TCP data by tcps command and displays setting contents
(2) Sets the display of log related to TCP protocol operation, and displays setting contents
(3) Sets the display of data length in tcpr (receiving data output display) / tcpsd (transmission data completion display) message and displays setting contents
(4) Sets the display of port number in tcpr (receiving data output display) / tcpsd (transmission data completion display) message and displays setting contents
(5) Sets the display of tcpsd to be the transmission completion display to the user by tcps command and displays setting contents
(6) Sets the default transmission TCP-Server port when sending by tcps command and displays setting contents
(7) Sets the TCP-Listen port so that TCP-Connection can be established on the specified TCP-Server port and displays setting contents
(8) Sets non-communication monitoring time after TCP-Connection is established and displays setting contents (in minutes)
(9) Sets the initial value of TCP packet re-transmission timer (in seconds)
(10) Sets the number of TCP packet re-transmissions
(11) Sets the number of SYN re-transmissions
(12) Sets the TCP MSS (Maximum Segment Size
(13) Sets the operation log display of TCP protocol

Command-line format

```
tcpopts <option> <enable|port|minutes|sec|num|bytes>
```

JSON command format

```
{ "tcpopts" : null }
{ "tcpopts" : { "option" : <stiring>, "enable" : <bool> } }
{ "tcpopts" : { "option" : <stiring>, "port" : <number> } }
{ "tcpopts" : { "option" : <stiring>, "minutes" : <number> } }
{ "tcpopts" : { "option" : <stiring>, "sec" : <number> } }
{ "tcpopts" : { "option" : <stiring>, "num" : <number> } }
{ "tcpopts" : { "option" : <stiring>, "bytes" : <number> } }
```

Command argument

Set parameters from argument “option” are set with argument “enable” or “port”.

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
option	string	auto_ connect	enable=1	To set the propriety setting for TCP-Connection automatic connection operation during TCP data transmission by command (3.3.6(3) tcps). “When this parameter is valid (true / on: 1), the connection of TCP-Connection is automatically established when data transmission by tcps.” and “When this parameter is valid (false / off: 0), TCP-Connection will not be connected by tcps, and TCP data will be transmitted by tcps command after TCP-Connection is established by “3.3.2(14) tcpcon” command.” (The set value for display of this option is set by enable value below.)"
		log	enable=0	To set the TCP protocol operation log display (The set value for display of this option is set by enable value below.)
		disp_len	enable=0	To set the transmission / reception data length display for reception display (tcpr) / transmission completion display (tcpsd). (The set value for display of this option is set by enable value below.)

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
		disp_port	enable=0	To set the transmission / reception port number display for reception display (tcpr) / transmission completion display (tcpsd). (The set value for display of this option is set by enable value below.)
		send_done	enable=0	To set the display of the transmission completion message (tcpsd) to the user by tcps command (The set value for the display of this option is set by the value of enable below.)
		send_port	port=3610 (Echonet-Lite)	To set the default transmission TCP-Server port number when the argument port is omitted by tcpcon / tcpdis / tcps command. (The set value for display of this option is set by port value below.)
		listen_ port	port=3610 (Echonet-Lite)	To set the TCP-Server port to be in the Listen state so that TCP-Connection can be established. (The set value for display of this option is set by port value below.) The maximum number of ports that can be registered is 4 ports.
option	string	idle_minutes	minutes=3 (TCP-connection disconnects after 3 minutes non-communication time expires)	To set the non-communication monitoring time (in minutes) after connection. The TCP-Connection disconnects automatically when the time during which TCP data is not sent or received expires for the minute specified by this parameter after establishing TCP-Connection by tcpcon / tcps command. When this parameter is set to 0, the non-communication monitoring will not work and the TCO-Connection will not be automatically disconnected. (The Set Value for this option is set by min value below.)
		rto_sec	sec=10	To set the Initial Value (in seconds) of the re-transmission timer (The set value of this option is set by sec value below.)
		maxrtx	num=3	To set the number of re-transmissions (When 0 is specified, there is no resend) (The set value of this option is set by num value below.)
		syn_maxrtx	num=5	Set the number of SYN re-transmission (When 0 is specified, there is no resend) (The set value of this option is set by num value below.)
		mss	bytes=536	To set the display for TCP MSS (Maximum Segment Size) (The Set Value for this option is set by bytes value below.)
enable	bool	0 (false) or 1 (true)	-	To set the enable / disable settings for auto_connect, log, send_done, disp_len, disp_port and disp_rssi of the above argument option
port	number	PortNumber (1-65534) or (-1)		To set the port number for listen_port and send_port of the above argument option. Deletes all the ports currently registered in listen_port when listen_port is specified in the above argument option and port number is set to -1 (*1)
minutes	number	Minutes (0-65535)		To set the time in minutes for idle_minutes in the above argument option
sec	number	Seconds (1-32)		Set the time in seconds for rto_sec in the above argument option
num	number	No.oftrials (0-12)		To set the number of trials for maxrtx and syn_maxrtx in the above argument option
bytes	number	Byte count (536-024)		Set the number of bytes for mss in the above argument option

*1: All ports are cancelled in Listen state and shifts to non-Listen state. All connected ports are disconnected.
For the disconnected port that has transitioned to the non-Listen state, the TIME_WAIT state is maintained internally for about 120 seconds. During this time, if Listen is set again on the same port, setting to the Listen state may fail due to the TCP "REUSE address". As a result, command error may be displayed when re-setting Listen to the same port by listen_port in the above argument option of tcpopts command. (Listen setting can be done again about 120 seconds after disconnection.)
If the argument is omitted, settings related to TCP data transmission / reception by the user will be displayed below.
If the value of command name is set to null in JSON command, the settings related to the TCP data transmission / reception contents by the user will be displayed below.

Command-line display format

Values described in "**Display format contents**" below are displayed in the following format as settings related to the TCP data transmission / reception contents from current user.

```
tcpopts auto_connect=<auto_connect> log=<log>
tcpopts disp_len=<disp_len> disp_port=<disp_port> send_done=<send_done>
tcpopts send_port=<send_port>
tcpopts listen_port=<listen_port1>,<listen_port2>,<listen_port3>,<listen_port4>
tcpopts idle_minutes=<idle_minutes> rto_sec=<rto_sec> maxrtx=<maxrtx> syn_maxrtx=<syn_maxrtx> mss=<syn_m
axrtx>
```

JSON command display format

Values described in "**Display format contents**" below are displayed in the following format as settings related to TCP data transmission / reception contents from user.

```
{ "udpopts" : { "comp_hdr" : <bool>, "comp_csum" : <bool>, "disp_len" : <bool>, "disp_port" : <bool>, "di
sp_rssi" : <bool>, "send_done" : <bool>, "send_port" : <number>, "listen_port1" : <number>, "listen_port
2" : <number>, "listen_port3" : <number>, "listen_port4" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
auto_connect	bool	on or off true or false in JSON command	To display the settings for the non-communication monitoring time (in minutes) after connection.
log			To display the settings of the TCP protocol operation log.
disp_len			To display the setting for displaying the transmission / reception data length in the reception (tcpr) / transmission completion display (tcpsd). (In JSON format, the transmission / reception data length is always displayed.)
disp_port			To display the setting for displaying the transmission / reception port number in the reception display (tcpr) / transmission completion display (tcpsd) (In JSON format, the transmission / reception port number is always displayed.)
send_done			To display the setting for the transmission completion (tcpsd) display to the user by tcps command.
send_port	number	port number (1~65534)	To display the default outbound TCP-Server port settings by tcps command.
listen_port1			To display the TCP-Server port that is in the Listen state so that TCP-Connection can be established.
listen_port2			
listen_port3			
listen_port4			
idle_minutes		minutes (1~65534)	To display the non-communication monitoring time (in minutes) after connection.
rto_sec		Seconds (1~32)	To display the initial value (in seconds) of the re-transmission timer.
maxrtx		numberof trials (0~12)	To display the setting of the number of re-transmissions. (When 0 is specified, there is no resend)
syn_maxrtx			To display the SYN re-transmission count setting. (When 0 is specified, there is no resend)
mss		byte count (536~1024)	To display TCP MSS (Maximum Segment Size).

Example of command line usage

<To show the setting value>

```
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
>
```

<To turn ON or OFF the automatic connection with tcps command>

```
>tcpopts auto_connect off
>tcps 2001:db8::1 0011223344556677
tcps:  no connection the TCP port.
>
>tcpopts auto_connect on
>tcpopts send_done on
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1>
>
```

<To turn ON the display of TCP protocol operation log>

```
>tcpopts log on
>tcpopts auto_connect on
>tcpopts send_done on
>tcps 2001:db8::1 0011223344556677
tcp_bind: bind to port 49153
tcp_connect to port 3610
State: SYN_SENT
State: ESTABLISHED
tcpsd <2001:db8::1>
>
```

<To turn ON or OFF the transmission/reception data length (disp_len)>

```
>tcpopts send_done on
>tcpopts disp_len off
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1>
>
>tcpopts send_done on
>tcpopts disp_len on
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (8)
>
```

<To turn ON or OFF the display of transmission/reception port (disp_port)>

```
>tcpopts send_done on
>tcpopts disp_len  off
>tcpopts disp_port off
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1>
>
>tcpopts send_done on
>tcpopts disp_len  off
>tcpopts disp_port on
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (3610)
>
```

<To set the Listen port (listen_port) so that the specified TCP port can be displayed in the received data display message (tcpr)>

```
>tcpopts disp_len  off
>tcpopts disp_rssi off
tcpopts: invalid parameter.
>tcpopts disp_port on
>tcpopts listen_port 5001
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=on send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610,5001
tcpopts idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
>
tcpr <2001:db8::2> (3610) 0011223344556677
>
```

<To set the default send TCP-Server port number (send_port) with the tcps command>

```
>tcpopts disp_port on
>tcpopts send_done on
>tcpopts send_port 5001
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=on send_done=on
tcpopts send_port=5001
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
>
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (5001)
>
```

<To set the non-communication monitoring time (in minutes) after connection>

```
>tcpopts idle_minutes 1
>tcpopts log on
>tcpopts send_done on
>tcps 2001:db8::1 0011223344556677
tcp_bind: bind to port 49156
tcp_connect to port 5001
State: SYN_SENT
State: ESTABLISHED
tcpsd <2001:db8::1> (5001)
>tcpstat
tcpstat num:2
tcpstat LISTEN      <NONE>             51877   <NONE>         3610
tcpstat FIN_WAIT_1  <2001:db8::1>       5001    <2001:db8::2>   49156
>
[ 1 minute later ]
tcp_close: closing in State: ESTABLISHED
State: FIN_WAIT_1
State: FIN_WAIT_2
tcp_slowtmr: removing pcb stuck in FIN-WAIT-2
tcp_pcb_purge
>tcpstat
tcpstat num:1
tcpstat LISTEN      <NONE>             51877   <NONE>         3610
>
```

<To set the initial value of the retransmission timer>

```
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
>
>tcpopts rto_sec 15
>
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=3 syn_maxrtx=5 mss=536
>
```

<To set the number of retransmissions>

```
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=[0]3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=3 syn_maxrtx=5 mss=536
>
>tcpopts maxrtx 7
>
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=7 syn_maxrtx=5 mss=536
>
```

<To set the SYN retransmission count>

```
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=7 syn_maxrtx=5 mss=536
>
>tcpopts syn_maxrtx 2
>
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=7 syn_maxrtx=2 mss=536
>
```

<To set the TCP MSS (Maximum Segment Size)>

```
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=[0]3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=7 syn_maxrtx=2 mss=536
>
>tcpopts mss 1000
>
>tcpopts
tcpopts auto_connect=on log=off
tcpopts disp_len=off disp_port=off send_done=off
tcpopts send_port=3610
tcpopts listen_port=3610
tcpopts idle_minutes=3 rto_sec=15 maxrtx=7 syn_maxrtx=2 mss=1000
>
```

Example of JSON command usage

<To show the setting value>

```
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 10, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
```

<To turn ON or OFF the automatic connection with tcps command>

```
>{ "tcpopts" : { "option" :  "auto_connect", "enable" : false } }
>{ "tcps" : { "ADDR" : "<2001:db8::1>", "data" : "0011223344556677" } }
{ "err" : { "cp" : 160, "cpnm" : "WSNCmd", "en" : -79, "desc" : "no connection the TCP port" } }
>
>{ "tcpopts" : { "option" :  "auto_connect", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcps" : { "ADDR" : "<2001:db8::1>", "data" : "0011223344556677" } }
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8 } }
>
```

<To turn ON the display of TCP protocol operation log>

```
>{ "tcpopts" : { "option" :  "log", "enable" : true } }
>{ "tcpopts" : { "option" :  "auto_connect", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcps" : { "ADDR" : "<2001:db8::1>", "data" : "0011223344556677" } }
tcp_bind: bind to port 49153
tcp_connect to port 3610
State: SYN_SENT
State: ESTABLISHED
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8 } }
>
```

<To set the Listen port (listen_port) so that the specified TCP port can be displayed in the received data display message (tcpr)>

```
>{ "json" : true }
{ "json" : true }
>
>{ "tcpopts" : { "option" :  "disp_len", "enable" : false } }
>{ "tcpopts" : { "option" :  "disp_len", "enable" : false } }
>{ "tcpopts" : { "option" :  "listen_port", "port" : 5001 } }
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 5000, "listen_port3" : 5001, "list
en_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 10, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
{ "tcpr" : { "ADDR" : "2001:db8::2", "port" : 5001, "len" : 8, "data" : "0011223344556677" } }
>
```

<To set the default send TCP-Server port number (send_port) with the tcps command>

```
>{ "tcpopts" : { "option" :  "disp_port", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_port", "port" : 5001 } }
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : true, "send_done"
 : true, "send_port" : 5001, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "list
en_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 10, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
>{ "tcps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 5001, "len" : 8 } }
>
```

<To set the non-communication monitoring time (in minutes) after connection>

```
>{ "tcpopts" : { "option" :  "send_port", "port" : 3610 } }
>{ "tcpopts" : { "option" :  "idle_minutes", "minutes" : 1 } }
>{ "tcpopts" : { "option" :  "log", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
tcp_bind: bind to port 49153
tcp_connect to port 3610
State: SYN_SENT
State: ESTABLISHED
>
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8 } }
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 2 } }
{ "tcpstat" : { "tcp_state" : "FIN_WAIT_1", "REMOTE_ADDR" : "2001:db8::1", "remote_port" : 3610, "LOCAL_
ADDR" : "2001:db8::2", "local_port" : 49153 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 37657, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
[ 1 minute later ]
tcp_close: closing in State: ESTABLISHED
State: FIN_WAIT_1
TCP connection closed: FIN_WAIT_1 3610 -> 49153.
tcp_pcb_purge

>{ "tcpstat" : null }
unknown command ({ "tcpstat" : null }).
>{ "tcpstat" : null }
{ "tcpstat" : { "num" : 1 } }
{ "tcpstat" : { "tcp_state" : "LISTEN", "REMOTE_ADDR" : "NONE", "remote_port" : 37657, "LOCAL_ADDR" : "NO
NE", "local_port" : 3610 } }
>
```

<To set the initial value of the retransmission timer>

```
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 10, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
>{ "tcpopts" : { "option" :  "rto_sec", "sec" : 15 } }
>
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
```

<To set the number of retransmissions>

```
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 3, "syn_maxrtx" : 5, "mss" : 536 } }
>
>{ "tcpopts" : { "option" :  "maxrtx", "num" : 7 } }
>
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 7, "syn_maxrtx" : 5, "mss" : 536 } }
>
```

<To set the SYN retransmission count>

```
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 7, "syn_maxrtx" : 5, "mss" : 536 } }
>
>{ "tcpopts" : { "option" :  "syn_maxrtx", "num" : 2 } }
>
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 7, "syn_maxrtx" : 2, "mss" : 536 } }
>
```

<To set the TCP MSS (Maximum Segment Size)>

```
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 7, "syn_maxrtx" : 2, "mss" : 536 } }
>
>{ "tcpopts" : { "option" :  "mss", "bytes" : 1000 } }
>
>{ "tcpopts" : null }
{ "tcpopts" : { "auto_connect" : true, "log" : false, "disp_len" : false, "disp_port" : false, "send_done
" : false, "send_port" : 3610, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "li
sten_port4" : 65535, "idle_minutes" : 3, "rto_sec" : 15, "maxrtx" : 7, "syn_maxrtx" : 2, "mss" : 1000 } }
>
```

3.3.8 Parameter commands

(1) param

Overview

This command is to display the set value of each command parameter saved in the non-volatile area.
This command does not support JSON format.
The reading of each value set in the non-volatile area is displayed using commands "3.3.8(2) save" and "3.3.8(4) svrst".
If commands "3.3.8(2) save" and "3.3.8(4) svrst" have not been executed, each value will be displayed as default value.

Command-line format

```
param
```

Command argument

Nothing

Command-line display format

Values described in the "**Display format contents**" below are displayed in the following format as the set value of parameters that can be set (saved in the non-volatile area) in the Wi-SUN module.
For the value displayed as "(*)" for each value, the default value is displayed instead of the value set in the non-volatile area.

```
    mac address : <MAC64B>
         pan id : <PAN16B>
   profile mode : <profile(name)>
      auth mode : <num>
UART baudrate : <rate>
   IPv6 address : GBL<IP128BGBL/prefix>
   DHCPv6 range : <range>
DCHPv6 fixed IP : <MAC64B> <IP128BGBL>
     mac-filter : default( <default> )
      ( <role> ): <MAC64B>
  Joiner-filter : default( <default> )
      ( <role> ): <MAC64B>
CCA threshold : <rssi>
antenna switch : <num>
radio FEC mode : <enable>
chrate : <rate>
        channel : low(<low>)<->high(<high>)num=<chnum>
   network name : <name>
       tcp_opts : [auto_connect] [send_done] [disp_len] [disp_port] [disp_rssi] send_port=<send_port>
                : listen_port=[listen_port1] [listen_port2] [listen_port3] [listen_port4]
                : idle_minutes=<idle_minutes> rto_sec=<rto_sec> maxrtx=<maxrtx> syn_maxrtx=<syn_maxrtx> m
ss=<mss>
       udp_opts : [send_done] [disp_len] [disp_port] [disp_rssi] send_port=<send_port> send_port_text=<se
nd_port_text>
                : listen_port=[listen_port1] [listen_port2] [listen_port3] [listen_port4]
                : listen_port_text=[listen_port_text1] [listen_port_text2] [listen_port_text3] [listen_po
rt_text4]
    rmtctl_opts : [send_done]
auto start role : <role_num(role_name)>
```

Display format contents

Display Parameter Name	Argument Name	Display Value	Description
mac address	MAC64B	<EUI64MACaddress>	To display the MAC address in EUI64 format
pan id	PAN16B	<PAN-ID>	To display the PAN-ID
profile mode	profile(name)	profile : 0 or 1	To display the number that corresponds to the profile mode name by the name below
		name:"NONE" or "FAN"	To display the profile mode name that corresponds to the above profile number
auth mode	num	0	Unauthenticated security mode
		1	Authenticated security mode

Display Parameter Name	Argument Name	Display Value	Description
UART baudrate	rate	2400/4800/9600/14400/19200/ 38400/57600/115200/230400/460800	Serial communication speed with command application
IPv6 address	IP128BGBL/ prefix	IP128BGBL: <IPv6 address>	To display the IPv6 global address
		prefix: prefix length	To display the prefix length of IPv6 global addresses
DHCPv6 range	Range	>1	To display the number of ranges from the minimum value to the maximum value of IPv6 address by DHCP payout
DCHPv6 fixed IP	MAC64B	<EUI64MAC address>	To display the EUI64 format MAC address associated with IPv6 address by fixed payout
	IP128BGBL	<IPv6 address>	To display the IPv6 address that can be fixed payout for the abovementioned <MAC64B>
mac-filter	default	"allow"	Permission setting. Receives and notifies incoming packets without filtering on MAC layer All incoming packets other than "deny" (= in denied setting) in <role> of <MAC64B> below are received and notified.
		"deny"	Denied setting. Filters incoming packets on MAC layer All incoming packets other than "allow" (= in permission setting) of <role> of <MAC64B> below are filtered.
	role	"allow"	Permission setting. Receives and notifies incoming packets of <MAC64B> below without filtering on MAC layer
		"deny"	Denied setting. Filters incoming packets of <MAC64B> with MAC layer.
	MAC64B	<EUI64MAC address>	The terminal MAC address to which the above <role> is applied.
joiner-filter	default	"allow"	Permission settings. Receives and notifies incoming packets on EAPOL layer inside the stack by nodef command without filtering All incoming packets other than "deny" (= in denied setting) in <role> of <MAC64B> below are received and notified.
		"deny"	Denied setting. Filters incoming packet on EAPOL layer in the stack by nodef command All incoming packets other than "allow" (= permission is set) in <role> of <MAC64B> below are filtered with MAC layer.
	role	"allow"	Permission settings. Receives and notifies incoming packets of <MAC64B> below by nodef command without filtering on EAPOL layer inside the stack
		"deny"	Denied setting. Filters incoming packets of <MAC64B> below by nodef command at the EAPOL layer inside the stack
	MAC64B	<EUI64MAC address>	The terminal MAC address to which the above <role> is applied
CCA threshold	rssi	-128~127	To display the RSSI threshold (-128 to 127) in CCA (Clear Channel Assessment) operation of PHY (RADIO)
chrate	rate	50/100/150/300	To display the current set value of the PHY transmission rate in Kbps
channel	low	0~128	To display the start-of-use channel number

Display Parameter Name	Argument Name	Display Value	Description
	high		To display the end-of-use channel number
	chnum		To display the number of channels used
network name	name	ASCII character string	To display the FAN network ID
tcp_opts	auto_connect	Argument name on the left when on (enable) Argument name on the left does not display when off (disable)	To display the settings for the non-communication monitoring time (in minutes) after connection
	log		To display the settings of the TCP protocol operation log
	send_done		To display the setting for displaying the transmission / reception data length in the reception (tcpr) / transmission completion display (tcpsd) (In JSON format, the transmission / reception data length is always displayed.)
	disp_len		To display the setting for displaying the transmission / reception port number in the reception display (tcpr) / transmission completion display (tcpsd). (In JSON format, the transmission / reception port number is always displayed.)
	disp_port		To display the setting for the transmission completion (tcpsd) display by tcps command
	send_port	port Number (1~65534)	To display the default outbound TCP-Server port settings in tcps command
	listen_port1		To display the TCP-Server port that is in the Listen state so that TCP-Connection can be established
	listen_port2		
	listen_port3		
	listen_port4		
	idle_minutes	minutes (1~65534)	To display the non-communication monitoring time (in minutes) after connection
	rto_sec	seconds (1~32)	To display the Initial Value (in seconds) of re-transmission timer
	maxrtx	no. of trials (0~12)	To display the setting of the number of re-transmissions (If 0 is specified, there will be no resend.)
	syn_maxrtx		To display the SYN re-transmission count setting (If 0 is specified, there will be no resend.)
	mss	byte count(536~1024)	To display TCP MSS (Maximum Segment Size)
udp_opts	disp_len	Argument name on the left when on (enable) Argument name on the left does not display when off (disable)	To display the setting for displaying the transmission / reception data length in the reception (udpr) / transmission completion display (udpsd)
	disp_port		To display the setting for displaying the transmission / reception port number in the reception (udpr) / transmission completion display (udpsd)
	disp_rssi		To display the setting for the display of received signal strength (RSSI) in the reception display (udpr)
	send_done		To display the setting for the display of transmission completion (udpsd) by udps command
	send_port	port number (1~65534)	To display the default outbound port settings by udps command
	send_port_text		To display the default outbound port settings for the udpst command

Display Parameter Name	Argument Name	Display Value	Description
	listen_port		To display the specified UDP ports as Listen port from 1 to 4 that enable reception by listen_port of udpopts command
	listen_port_text		To display the specified UDP ports as Listen port from 1 to 4 that enable reception by listen_port_text of udpopts command
rmtctl_opts	send_done	Argument name on the left when on (enable) Argument name on the left does not display when off (disable)	To display the setting for the display of the transmission completion (rmtsd) by rmtcmd command
auto start role	role_num (role_name)	role_num : 0~3	To display the role number that corresponds to the role (terminal operation mode) base on <role_name> below 0: Non-operation mode 1: Border Router terminal operation mode 2: Router terminal operation mode 3: Leaf terminal operation mode
		role_name : "NONE" "BORDER" "ROUTER" "LEAF"	To display the role (terminal operation mode) name that corresponds to the number <role_num> above "NONE": Non-operation mode "BORDER": Border Router terminal operation mode "ROUTER": Router terminal operation mode "LEAF": Leaf terminal operation mode

Example of command line usage

```
>param
    mac address : <0001020304050607>(*)
         pan id : <NONE>
   profile mode : 1(FAN)
      auth mode : 1
   IPv6 address : GBL<NONE/64>
   DHCPv6 range : 1000
DHCPv6 fixed IP : (*)
     mac-filter : default( allow )
  joiner-filter : default( allow )
  CCA threshold : -84
radio FEC mode : 0
antenna switch : 1(*)
       chrate  : 150Kbps
       channel : low(33)<->high(59),num=14
   network name : Wi-SUN-FAN
    TCP options : auto_connect disp_len disp_port send_done send_port=3610
                : listen_port=3610
                : idle_minutes=3 rto_sec=10 maxrtx=3 syn_maxrtx=5 mss=536
    UDP options : disp_len disp_port disp_rssi send_port=3610 send_port_text=20171
                : listen_port=3610
                : listen_port_text=20171
 RMTCTL options : send_done
auto start role : 0(NONE)
>
```

(2) save

Overview

Set value of each command parameter is saved in the non-volatile area.
The current value of the parameter that can be saved with "3.3.8(1) param" command is saved in a non-volatile memory such as Flash by this command.

Command-line format

```
save
```

JSON command format

```
{ "save" : null }
```

Command argument

None

Command-line display format

The string "save parameter is saved" is displayed.

JSON command display format

The completion message will be displayed in the JSON format such as follows.

```
{ "save" : "parameter is saved" }
```

Example of command line usage

```
>save
save parameter is saved
>
```

Example of JSON command usage

```
>{ "save" : null }
{ "save" : "parameter is saved" }
>
```

(3) clear

Overview

Every value saved for each command parameter in the non-volatile area is saved.
The setting of non-volatile memory such as Flash that stores the parameters that can be displayed by “3.3.8(1) param” command is returned to initial value by this command.
When clear command is issued, the corresponding parameter will not change and only the set value of the non-volatile memory is initialized.

Command-line format

clear

JSON command format

```
{ "clear" : null }
```

Command argument

None

Command-line display format

The string "clear parameter is cleared" is displayed.

JSON command display format

The completion message will be displayed in the JSON format below.

```
{ "clear" : "parameter is cleared" }
```

Command-line application example

```
>clear
clear parameter is cleared
>
```

JSON command application example

```
>{ "clear" : null }
{ "clear" : "parameter is cleared" }
>
```

Example of command line usage

```
>clear
clear parameter is cleared
>
```

Example of JSON command usage

```
>{ "clear" : null }
{ "clear" : "parameter is cleared" }
>
```

(4) svrst

Overview

The set value of each command parameter is saved in the non-volatile area and a system reset (reboot) is executed.
(An equivalent operation is performed by combining commands "3.3.8(2) save" and "3.3.1(4) reset".)

Command-line format

```
svrst [delay_sec]
```

JSON command format

```
{ "svrst" : null }
{ "svrst" : <number> }
{ "svrst" : { "delay_sec" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
delay_sec	number	0~3600	System reset execution delay time (in seconds)

If the argument is omitted in the above command-line format, the system reset will be executed immediately with a delay time of 0 (seconds).
If the value of command name is set to null in JSON command, the system reset will be executed immediately with a delay time of 0 (seconds).

Command-line display format

Values described in **Display format contents** below are displayed in the following format.

```
svrst parameter is saved and reset delay <delay_sec>sec
```

JSON command display format

Values described in **Display format contents** below are displayed in the following format.

```
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
delay_sec	number	>0 (input value)	System reset execution delay time (in seconds)
desc	string	(character string on the right)	The operation summary string "parameter is saved and reset" is displayed.

Example of command line usage

```
>svrst
svrst parameter is saved and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>svrst 5
svrst parameter is saved and reset delay 5sec
>[~~~ 5 seconds later ~~~]
inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
```

Example of JSON command usage

```
>{ "svrst" : null }
{ "svrst" : { "delay_sec" : 0, "desc" : "parameter is saved and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
>{ "svrst" : { "delay_sec" : 5 } }
{ "svrst" : { "delay_sec" : 5, "desc" : "parameter is saved and reset" } }
>[~~~ 5 seconds later ~~~]
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
```

(5) clrst

Overview

Every value of each command parameter saved in the non-volatile area is cleared and a system reset (reboot) is executed.
(The equivalent operation is performed by combining commands "3.3.8(3) clear" and "3.3.1(4) reset".)

Command-line format

```
clrst [delay_sec]
```

JSON command format

```
{ "clrst" : null }
{ "clrst" : <number> }
{ "clrst" : { "delay_sec" : <number> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
delay_sec	number	0~3600	System reset execution delay time (in seconds)

If the argument is omitted in the above command-line format, the system reset will be executed immediately with a delay time of 0 (seconds).
If the value of the command name is set to null in JSON command, the system reset will be executed immediately with a delay time of 0 (seconds).

Command-line display format

Values described in **Display format contents** below are displayed in the following format.

```
clrst parameter is cleared and reset delay <delay_sec>sec
```

JSON command display format

Values described in **Display format contents** below are displayed in the following format.

```
{ "clrst" : { "delay_sec" : 0, "desc" : "parameter is cleared and reset" } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
delay_sec	number	>0 (inputvalue)	System reset execution delay time (in seconds)
desc	string	(character string on the right)	To display the operation summary character string "parameter is cleared and reset"

Example of command line usage

```
>clrst
clrst parameter is cleared and reset delay 0sec
>inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
>clrst 5
clrst parameter is cleared and reset delay 5sec
>[~~~ 5 seconds later ~~~]
inf 01,01,0,0      {    WSN: system booted.                    }
////////////////////////////////////////////////////////////
// Copyright (C) 2020 Nissin Systems Co.,Ltd.
// EW-WSN-FAN-1.0.56.60 linux(x86_64:socket)
// Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)
////////////////////////////////////////////////////////////
init 0(NONE)
>
```

Example of JSON command usage

```
>{ "clrst" : null }
{ "clrst" : { "delay_sec" : 0, "desc" : "parameter is cleared and reset" } }
>{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
>{ "clrst" : { "delay_sec" : 5 } }
{ "clrst" : { "delay_sec" : 5, "desc" : "parameter is cleared and reset" } }
>[~~~ 5 seconds later~~~]
{ "inf" : { "cp" : 1, "cpnm" : "WSN", "en" : 1, "desc" : "system booted." } }
{ "vers" : { "Copyright" : "Copyright (C) 2020 Nissin Systems Co.,Ltd.", "FirmwareBSP" : "EW-WSN-FAN-1.0.
56.60 linux(x86_64:socket)", "ProfileBuild" : "Wi-SUN Profile for FAN (Sep  2 2020 12:44:31)" } }
{ "init" : "0(NONE)" }
>
```

3.3.9 Remote commands

(1) rmtcmd

Overview

The command statement implemented by this command application is sent as message to a remote terminal connected to the network and on the remote terminal that received the message, the command statement is executed.
As for the message of command execution result (content of command execution display) on the remote terminal that received the message, a notification is displayed as execution result of this command in "4.3.3(1) rmtmsg".
If an error related to the remote command message itself occurs with this command, the error content is displayed in "4.3.3(3) rmterr".
Sending and receiving messages to and from remote terminals and executing commands with this command shall be limited to the terminal on which this command application is mounted.
Port number 48878 is used for UDP communication to send and receive messages by rmtcmd command and “4.3.3(1) rmtmsg” command.

Command-line format

```
rmtcmd <ADDR> <cmd>
```

JSON command format

```
{ "rmtcmd" : { "ADDR" : <string>, "cmd" : <JSON Object> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Description
ADDR	string	<IPv6 address>	Destination IPv6 unicast address (Not supported in IPv6 multicast addresses)
cmd	JSON Object	execution command statement	Command statement to be executed on the remote terminal Execution command statement in the command-line format: <command name> [argument 1] [argument 2] For the execution command statement, enter the command statement according to each command format to be executed. Execution command statement in JSON format: (Execution statement without command arguments): { "<command name>" : null } (Execution statement with command arguments): {"<command name>": <argument 1 value>} {"<command name>": {"argument 1 name": <argument 1 value>,…}} For execution command statement by JSON command, the character string for execution command statement is entered as the value of JSON Object for "cmd" which is the JSON format of rmtcmd mentioned above according to the JSON command format for each command to be executed.

As for the result of remote command execution, a notification is displayed as the execution result of this command in "4.3.3(1) rmtmsg".
Refer to "4.3.3(1) rmtmsg" for the message display of execution result.

Example of command line usage

<To execute the mac command on a remote terminal and get the MAC address of the remote terminal>

```
>rplsr
rplsr - Routing links (1 in total):
rplsr -- 2001:db8::2 to 2001:db8::1 (lifetime: 7151 seconds)>mac
>
>mac
mac <0001020304050601>,LLA<fe80::201:203:405:601>
>
>rmtcmd 2001:db8::2 mac
rmtmsg <2001:db8::2>: mac <0001020304050602>,LLA<fe80::201:203:405:602>
>
```

<To operate the MAC filter table on a remote terminal with the macf command>

```
>rmtcmd 2001:db8::2 macf
rmtmsg <2001:db8::2>: macf default             ( deny  )
rmtmsg <2001:db8::2>: macf <0001020304050601>   ( allow )
rmtmsg <2001:db8::2>: macf <0001020304050603>   ( allow )
>
>rmtcmd 2001:db8::2 macf allow 0001020304050605
>
>rmtcmd 2001:db8::2 macf
rmtmsg <2001:db8::2>: macf default             ( deny  )
rmtmsg <2001:db8::2>: macf <0001020304050601>   ( allow )
rmtmsg <2001:db8::2>: macf <0001020304050603>   ( allow )
rmtmsg <2001:db8::2>: macf <0001020304050605>   ( allow )
>
```

<To send UDP data to the local terminal with the udps command on the remote terminal and receive it on the local terminal>

```
>udpopts listen_port 50001
>udpopts disp_port on
>udpopts
udpopts comp_hdr=on comp_csum=off
udpopts disp_len=off disp_port=on disp_rssi=off send_done=off
udpopts send_port=3610 send_port_text=20171
udpopts listen_port=3610,50001
udpopts listen_port_text=20171
>
>rmtcmd 2001:db8::2 udpopts send_port 50001
>rmtcmd 2001:db8::2 udpopts disp_port on
>rmtcmd 2001:db8::2 udpopts send_done on
>rmtcmd 2001:db8::2 udpopts
rmtmsg <2001:db8::2>: udpopts comp_hdr=on comp_csum=off
rmtmsg <2001:db8::2>: udpopts disp_len=off disp_port=on disp_rssi=off send_done=on
rmtmsg <2001:db8::2>: udpopts send_port=50001 send_port_text=20171
rmtmsg <2001:db8::2>: udpopts listen_port=3610
rmtmsg <2001:db8::2>: udpopts listen_port_text=20171
>
>rmtcmd 2001:db8::2 udps 2001:db8::1 0011223344556677
udpr <2001:db8::2> (50001) 0011223344556677
rmtmsg <2001:db8::2>: udpsd <2001:db8::1> (50001)
>
```

<To check that a ping reaches another terminal using the ping command on a remote terminal>

```
>rmtcmd 2001:db8::3 ping start 2001:db8::2 64 1 2 64
rmtmsg <2001:db8::3>: ping <2001:db8::2> (seq=1 sz=64bytes time=0.200sec) 1/64
rmtmsg <2001:db8::3>: ping <2001:db8::2> (seq=2 sz=64bytes time=0.050sec) 2/64
rmtmsg <2001:db8::3>: ping <2001:db8::2> (seq=3 sz=64bytes time=0.080sec) 3/64
>rmtcmd 2001:db8::3 ping state
rmtmsg <2001:db8::3>: 3 transmitted, 3 received, 0.0% loss (min=0.050/max=0.200/avr=0.110 sec)
rmtmsg <2001:db8::3>: ping <2001:db8::2> (seq=4 sz=64bytes time=0.080sec) 4/64
rmtmsg <2001:db8::3>: ping <2001:db8::2> (seq=5 sz=64bytes time=0.080sec) 5/64
>rmtcmd 2001:db8::3 ping stop
rmtmsg <2001:db8::3>: 5 transmitted, 5 received, 0.0% loss (min=0.050/max=0.200/avr=0.098 sec)
>
```

Example of JSON command usage

<To execute the mac command on a remote terminal and get the MAC address of the remote terminal>

```
>{ "rplsr" : null}
{ "rplsr" : { "num" : 1 } }
{ "rplsr" : { "IP128B" : "2001:db8::2", "IP128B_PARENT" : "2001:db8::1", "lifetime" : 7155 } }
>
>{ "mac" : null}
{ "mac" : { "MAC64B" : "0001020304050601", "IP128BLLA" : "fe80::201:203:405:601" } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "mac" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "mac" : { "MAC64B" : "0001020304050602", "IP128BLLA" : "
fe80::201:203:405:602" } } } }
>
```

<To operate the MAC filter table on a remote terminal with the macf command>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "macf" : null } } }
>
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "default" : "deny" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "000102030405060
1" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "000102030405060
3" } } } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "macf" : { "role" : "allow",  "MAC64B" : "000102030405
0605" } } } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "macf" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "default" : "deny" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "000102030405060
1" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "000102030405060
3" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "000102030405060
5" } } } }
>
```

<To send UDP data to the local terminal with the udps command on the remote terminal and receive it on the local terminal>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "listen_port", "port" : 5001 } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "udpopts" : { "option" :  "send_port", "port" : 5001 }
 } } }
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "udpopts" : { "option" :  "disp_port", "enable" : true
 } } } }
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "udpopts" : { "option" :  "send_done", "enable" : true
 } } } }
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "udpopts" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "udpopts" : { "comp_hdr" : true, "comp_csum" : false, "d
isp_len" : false, "disp_port" : true, "disp_rssi" : false, "send_done" : true, "send_port" : 5001, "send_
port_text" : 20171, "listen_port1" : 3610, "listen_port2" : 65535, "listen_port3" : 65535, "listen_port4
" : 65535, "listent_port_text1" : 20171, "listent_port_text2" : 65535, "listent_port_text3" : 65535, "li
stent_port_text4" : 65535 } } } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "udps" : { "ADDR" : "2001:db8::1", "data" : "001122334
4556677" } } } }
{ "udpr" : { "ADDR" : "2001:db8::2", "port" : 5001, "rssi" : -35, "len" : 8, "data" : "0011223344556677"
} }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "udpsd" : { "ADDR" : "2001:db8::1", "port" : 5001, "len"
 : 8 } } } }
>
```

<To check that a ping reaches another terminal using the ping command on a remote terminal>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "ping" : { "cmd" : "start", "ADDR" : "2001:db8::1", "s
ize" : 1024, "type" : 0, "sec" : 2, "cnt" : 64 } } } }
{ "inf" : { "cp" : 52, "cpnm" : "icmpEch", "en" : 50, "desc" : "rcvd echo request (seq=1 len=1024)" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 1, "size
" : 1024, "msec" : 200, "txcnt" : 1, "cnt" : 64, "rank" : 320 } } } }
{ "inf" : { "cp" : 52, "cpnm" : "icmpEch", "en" : 50, "desc" : "rcvd echo request (seq=2 len=1024)" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 2, "size
" : 1024, "msec" : 90, "txcnt" : 2, "cnt" : 64, "rank" : 320 } } } }
{ "inf" : { "cp" : 52, "cpnm" : "icmpEch", "en" : 50, "desc" : "rcvd echo request (seq=3 len=1024)" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 3, "size
" : 1024, "msec" : 100, "txcnt" : 3, "cnt" : 64, "rank" : 320 } } } }
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "ping" : { "cmd" : "state"} } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_state" : { "txcnt" : 3, "rxcnt" : 3, "min_msec" :
90, "max_msec" : 200, "avr_msec" : 130, "total_msec" : 390 } } } }
{ "inf" : { "cp" : 52, "cpnm" : "icmpEch", "en" : 50, "desc" : "rcvd echo request (seq=4 len=1024)" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 4, "size
" : 1024, "msec" : 40, "txcnt" : 4, "cnt" : 64, "rank" : 320 } } } }
{ "inf" : { "cp" : 52, "cpnm" : "icmpEch", "en" : 50, "desc" : "rcvd echo request (seq=5 len=1024)" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_recv" : { "ADDR" : "2001:db8::1", "seq" : 5, "size
" : 1024, "msec" : 100, "txcnt" : 5, "cnt" : 64, "rank" : 320 } } } }
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "ping" : { "cmd" : "stop"} } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "ping_state" : { "txcnt" : 5, "rxcnt" : 5, "min_msec" :
40, "max_msec" : 200, "avr_msec" : 106, "total_msec" : 530 } } } }
>
```

(2) rmtopts

Overview

This command displays the settings related to contents of sending and receiving remote command messages

For contents of sending and receiving remote command messages, the following settings and displays are carried out.
To set the display setting for rmtsd message that is the transmission / reception display to user and displays setting contents by "3.3.9(1) rmtcmd" command.

Command-line format

```
rmtopts <option> <enable>
```

JSON command format

```
{ "rmtopts" : null }
{ "rmtopts" : { "option" : <stiring>, "enable" : <bool> } }
```

Command argument

Argument Name	JSON Argument Type	Set Value	Initial Value	Description
option	string	send_done	enable=0	To set the display of transmission completion message (rmtsd) to the user by rmtcmd command

If the argument is omitted, the settings related to the transmission / reception contents of the remote command message will be displayed below.
If the value of the command name is null in JSON command, the settings related to sending and receiving remote command messages will be displayed below.

Command-line display format

Values described in **Display format contents** mentioned below will display the display setting for UDP data transmission / reception contents from user and settings related to the UDP packet format in the following format

```
udpopts send_done=<send_done>
```

JSON command display format

Values described in **Display format contents** mentioned below will display the display setting for UDP data transmission / reception contents from user and settings related to the UDP packet format in the following format.

```
{ "udpopts" : { "send_done" : <bool> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
send_done	bool	On or off true or false in JSON command	To display the setting for displaying transmission completion (udpsd) message to the user by udps command.

Example of command line usage

<To show the setting value>

```
>rmtopts
rmtopts send_done=on
>
```

<To turn ON or OFF the display of transmission completion (send_done)>

```
>rmtopts send_done off
>
>rmtcmd 2001:db8::2 mac
rmtmsg <2001:db8::2>: mac <0001020304050602>,LLA<fe80::201:203:405:602>
>
>rmtopts send_done on
>
>rmtcmd 2001:db8::2 mac
rmtsd <2001:db8::2> mac
rmtmsg <2001:db8::2>: mac <0001020304050602>,LLA<fe80::201:203:405:602>
>
```

Example of JSON command usage

<To show the setting value>

```
>{ "rmtopts" : null }
{ "rmtopts" : { "send_done" : false } }
>
```

<To turn ON or OFF the display of transmission completion (send_done)>

```
>{ "rmtopts" : { "option" : "send_done", "enable" : false } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "mac" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "mac" : { "MAC64B" : "0001020304050602", "IP128BLLA" : "
fe80::201:203:405:602" } } } }
>
>{ "rmtopts" : { "option" : "send_done", "enable" : true } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "mac" : null } } }
{ "rmtsd" : { "ADDR" : "2001:db8::2", "cmd" : "{ \"mac\" : null }" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "mac" : { "MAC64B" : "0001020304050602", "IP128BLLA" : "
fe80::201:203:405:602" } } } }
>
```

4. Message Output

4.1 Message output type classification

The message output contents are classified as follows.

No	Classification	Description
1	Received data output	Outputs when receiving various packets
2	Transmission completion output	Outputs when various packet transmissions are completed
3	Remote command message output	Outputs the command execution result when the command is executed on a remote terminal

4.2 Message output list

A list of message outputs is shown below.

4.2.1 Received data message output

No	Command	Description
1	udpr	UDP message reception (binary code reception)
2	udprt	UDP message reception (text data reception)
3	tcpr	TCP data reception
4	ping	Receive ICMP Echo response

4.2.2 Send completion message output

No	Command	Description
1	udpsd	UDP message transmission completion output
2	tcpsd	TCP data transmission complete output

4.2.3 Remote command message output

No	Command	Description
1	rmtmsg	Receive and output the execution result message on the remote terminal by rmtcmd command
2	rmtsd	Rmtcmd command transmission completion output
3	rmterr	Error message output related to sending and receiving messages of the remote command itself

4.3 Message output specifications

4.3.1 Received data message

(1) udpr

Overview

The message outputs when data is received by UDP protocol.
Reception port for UDP packets is 3610 by default. When receiving with the reception port set by the user, set the reception port with the listen_port option of "3.3.7(1) udpopts".
Displays in JSON format when JSON is enabled by "3.3.1(6) json" command.

Command-line format

```
udpr <ADDR> [([port]/[rssi])] [(len)]<data>
```

JSON command format

```
{ "udpr" : { "ADDR" : <string>, "port" : <number>, "rssi" : <number>, "len" : <number>, "data" : <string>
 } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the IPv6 address source of transmission.
port	number	1~65534	To display the UDP receiving port number (Normally not displayed on command-line. It will be displayed when the display setting is enabled in the "disp_port" option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
rssi	number	-128~127	To display the received signal strength (RSSI) when the said UDP packet is received (Not normally displayed on command-line. It will be displayed when the display setting is enabled in the "disp_rssi" option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
len	number	1~65535	To display the received data length (Not normally displayed on command-line. It will be displayed when the display setting is enabled with the "disp_len" option of "3.3.7(1) udpopts".) (Always displayed in JSON command.)
data	string	hexadecimal ASCII character string	To display the received data payload The data payload is displayed as a hexadecimal ASCII character string. Two characters make 1 byte of data.

Example of ccommand line usage

<Normal received data message output with default settings by udpopts command>

```
udpr <2001:db8::1> 0011223344556677
>
```

<Received data message output when port number output setting (disp_port) is turned ON by udpopts command>

```
>udpopts disp_port on
>udpopts disp_rssi off
>udpopts disp_len off
>
udpr <2001:db8::1> (3610) 0011223344556677
>
```

<Received data message output when RSSI output setting (disp_rssi) is turned ON by udpopts command>

```
>udpopts disp_rssi on
>udpopts disp_port off
>udpopts disp_len off
>
udpr <2001:db8::1> (-35) 0011223344556677
>
```

<Received data message output when port number (disp_port)/RSSI output setting (disp_rssi) is turned ON by udpopts command>

```
>udpopts disp_rssi on
>udpopts disp_port on
>udpopts disp_len off
>
udpr <2001:db8::1> (3610/-35) 0011223344556677
>
```

<Received data message output when received data length output (disp_len) is turned ON by the udpopts command>

```
>udpopts disp_len on
>udpopts disp_rssi off
>udpopts disp_port off
>
udpr <2001:db8::1> (8)0011223344556677
>
```

<Received data message output when port number/RSSI/received data length output setting is turned ON by udpopts command>

```
>udpopts disp_len on
>udpopts disp_rssi on
>udpopts disp_port on
>
udpr <2001:db8::1> (3610/-35) (8)0011223344556677
>
```

<Received data message output when the user setting port (listen_port) is set by the udpopts command>

```
>udpopts disp_port on
>udpopts disp_len off
>udpopts disp_rssi off
>udpopts listen_port 5001
>
udpr <2001:db8::1> (5001) 0011223344556677
>
```

Example of JSON command usage

<Received data JSON message output by json command>

```
>{ "json" : true }
{ "json" : true }
>
{ "udpr" : { "ADDR" : "2001:db8::1", "port" : 3610, "rssi" : 0, "len" : 8, "data" : "0011223344556677" }
}
>
```

<Received data message output when the user setting port (listen_port) is set by the udpopts command>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" : "listen_port", "port" : 5001 } }
>
{ "udpr" : { "ADDR" : "2001:db8::1", "port" : 3610, "rssi" : 0, "len" : 8, "data" : "0011223344556677" }
}
>
```

(2) udprt

Overview

The message outputs when data is received on the text reception port by UDP protocol.
The port for receiving UDP packets text is 20171 by default. When receiving with port for receiving text set by user, set the reception port by listen_port option of "3.3.7(1) udpopts".
Displays in JSON format when JSON is enabled by “3.3.1(6) json” command.

Command-line format

```
udprt <ADDR> [([port]/[rssi])] [(len)]"<data>"
```

JSON command format

```
{ "udprt" : { "ADDR" : <string>, "port" : <number>, "rssi" : <number>, "len" : <number>, "data" : <string
> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the IPv6 address of transmission source
port	number	1~65534	To display the UDP receiving port number (Not normally displayed on command-line. It will be displayed when the display setting is enabled with the “disp_port” of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
rssi	number	-128~127	To display the received signal strength (RSSI) when the UDP packet is received (Not normally displayed on command-line. It will be displayed when the display setting is enabled with the “disp_rssi” option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
len	number	1~65535	To display the UDP packet payload received data length. (Not normally displayed on command-line. It will be displayed when the display setting is enabled with the “disp_len” option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
data	string	ASCII character string	To display the UDP packet received data payload. Received data payload character string is displayed as literal character string enclosed by double quotation marks ('"'). (The double quotation marks ('"') of the string literal to be displayed are not included in the received UDP packet payload.) When displaying a character in which the control character represented by the escape character ('\') is included in the character string of the received data payload,the control character received in the payload of the UDP packet is converted to a 2-byte character string including the escape character ('\') and displayed. The payload is received and displayed up to 1024 Octets (received data is 1024 Bytes in ASCII).

Example of command line usage

<Normal received data message output with default settings by udpopts command>

```
udprt <2001:db8::1> "abcdef"
>
```

<Received data message output when port number output setting (disp_port) is turned ON by udpopts command>

```
>udpopts disp_port on
>udpopts disp_rssi off
>udpopts disp_len off
>
udprt <2001:db8::1> (20171) "abcdef"
>
```

<Received data message output when RSSI output setting (disp_rssi) is turned ON by udpopts command>

```
>udpopts disp_rssi on
>udpopts disp_port off
>udpopts disp_len off
>
udprt <2001:db8::1> (-35) "abcdef"
>
```

<Received data message output when received data length output (disp_len) is turned ON by udpopts command>

```
>udpopts disp_len on
>udpopts disp_rssi off
>udpopts disp_port off
>
udprt <2001:db8::1> (6)"abcdef"
>
```

<Received data message output when the text reception port (listen_port_text) is set by udpopts command>

```
>udpopts disp_port on
>udpopts disp_len off
>udpopts disp_rssi off
>udpopts listen_port_text 5001
>
udprt <2001:db8::1> (5001) "abcdef"
>
```

<Receive with space character (' ')>

```
>udpopts disp_len on
>
udprt <2001:db8::1> (9)"1234 5678"
>
```

<Receive with double quote ('\"')>

```
>udpopts disp_len on
>
udprt <2001:db8::1> (9)"1234\"5678"
>
```

<Receive with tab character ('\t') and CR newline character ('\n')>

```
>udpopts disp_len on
>
udprt <2001:db8::1> (10)"1234\t\n5678"
>
```

<Receive the string "\n">

```
>udpopts disp_len on
>
udprt <2001:db8::1> (10)"1234\t\n5678"
>
```

Example of JSON command usage

<Received data JSON message output by json command>

```
>{ "json" : true }
{ "json" : true }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 6, "data" : "abcdef" } }
>
```

<Received data message output when the text reception port (listen_port_text) is set by udpopts command>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" : "listen_port", "port" : 5001 } }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 6, "data" : "abcdef" } }
>
```

<Receive with space character (' ')>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_port", "enable" : false } }
>{ "udpopts" : { "option" :  "disp_rssi", "enable" : false } }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 9, "data" : "1234 5678" } }
```

<Receive with double quote ('\"')>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_port", "enable" : false } }
>{ "udpopts" : { "option" :  "disp_rssi", "enable" : false } }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 9, "data" : "1234\"5678" } }
>
```

<Receive with tab character ('\t') and CR newline character ('\n')>

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_port", "enable" : false } }
>{ "udpopts" : { "option" :  "disp_rssi", "enable" : false } }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 10, "data" : "1234\t\n5678" }
 }
>
```

<Receive the string "\n">

```
>{ "json" : true }
{ "json" : true }
>
>{ "udpopts" : { "option" :  "disp_len", "enable" : true } }
>{ "udpopts" : { "option" :  "disp_port", "enable" : false } }
>{ "udpopts" : { "option" :  "disp_rssi", "enable" : false } }
>
{ "udprt" : { "ADDR" : "2001:db8::1", "port" : 20171, "rssi" : -35, "len" : 10, "data" : "1234\\n5678" }
}
>
```

(3) tcpr

Overview

The message outputs when data is received using TCP protocol.
It displays in JSON format when JSON is enabled by “3.3.1(6) json” command.

Command-line format

```
tcpr <ADDR> [(port)] [(len)]<data>
```

JSON command format

```
{ "tcpr" : { "ADDR" : <string>, "port" : <number>, "len" : <number>, "data" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the IPv6 unicast address transmission source
port	number	1~65535	TCP-Server port number of TCP data received packet For the port number to be displayed, TCP-Connection is established and the TCP-Server port number used for the received packet is displayed. (Refer to overview of “3.3.2(14) tcpcon” command for "TCP-Server address" / "TCP-Server port".) (Not normally displayed on command-line. It will be displayed when the display setting is enabled in the "disp_port" option of “3.3.7(2) tcpopts”.) (Always displayed in JSON command.)
len	number	1~65535	To display the received data length (Not normally displayed on command-line. It will be displayed when the display setting is enabled in the "disp_len" option of “3.3.7(2) tcpopts”.) (Always displayed in JSON command.)
data	string	hexadecimal ASCII character string	To display the TCP packet received data payload The received data payload is displayed as a hexadecimal ASCII character string. Two characters make 1 byte of data.

Example of command line usage

<Normal received data message output at default setting by tcpopts command>

```
tcpr <2001:db8::1> 0011223344556677
>
```

<Received data message output when port number output setting (disp_port) is turned ON by tcpopts command>

```
>tcpopts disp_port on
>tcpopts disp_len off
>
tcpr <2001:db8::1> (3610) 0011223344556677
>
```

<Received data message output when received data length output (disp_len) is turned ON by tcpopts command>

```
>tcpopts disp_len on
>tcpopts disp_port off
>
tcpr <2001:db8::1> (8)0011223344556677
>
```

<Received data message output when port number/received data length output setting is turned ON by tcpopts command>

```
>tcpopts disp_len on
>tcpopts disp_port on
>
tcpr <2001:db8::1> (3610) (8)0011223344556677
>
```

<Received data message output when the user setting port (listen_port) is set by tcpopts command>

```
>tcpopts disp_port on
>tcpopts disp_len on
>tcpopts listen_port 5001
>
tcpr <2001:db8::1> (5001) (8)0011223344556677
>
```

Example of JSON command usage

<Received data JSON message output by json command>

```
>{ "json" : true }
{ "json" : true }
>
{ "tcpr" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8, "data" : "0011223344556677" } }
>
```

<Received data message output when the user setting port (listen_port) is set by tcpopts command>

```
>{ "json" : true }
{ "json" : true }
>
>{ "tcpopts" : { "option" : "listen_port", "port" : 5001 } }
>
{ "tcpr" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8, "data" : "0011223344556677" } }
>
```

(4) ping (ping_recv)

Overview

The message outputs when ICMPv6 ECHO Reply is received after the ICMPv6 ECHO Request Issuance by "3.3.6(4) ping" command.
If ICMPv6 ECHO Reply cannot be received from the other terminal, nothing is displayed.

Command-line display format

```
ping <ADDR> (seq=<seq> sz=<size>bytes time=<msec>sec)<txcnt>/<cnt>
```

JSON command display format

```
{ "ping_recv" : { "ADDR" : <string>, "seq" : <number>, "size" : <number>, "msec" : <number>, "txcnt" : <n
umber>, "cnt" : <number>, "rank" : <number> } }
```

Display format contents

(When receiving ICMPv6 ECHO Reply)

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the destination (ICMP Echo Reply Source) IPv6 address.
seq	number	0 or more	To display the sequence number.
size			To display the payload length (number of bytes).
msec			To display the response time (sec).
txcnt			To display the number of transmission completions.
cnt			To display the total number of scheduled transmissions.
rank			To display the local terminal RPL rank (displayed in JSON only) at the time of reception.

4.3.2 Transmission completion message

(1) udpsd

Overview

The message outputs when transmission of packet by "3.3.6(1) udps " command is completed.
This message will not be displayed at normal default settings.
Displayed when the transmission completion message display is enabled in send_done option of "3.3.7(1) udpopts"

Command-line display format

```
udpsd <ADDR>[(port)] [(len)]
```

JSON command display format

```
{ "udpsd" : { "ADDR" : <string>, "port" : <number>, "len" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the destination MAC address or IPv6 address.
port	number	1~65535	To display the UDP receiving port number (Not normally displayed on command-line. It will be displayed when the display setting is enabled by “disp_port” option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)
len	number	1~65535	To display the transmission data length (Not normally displayed on command-line. It will be displayed when the display setting is enabled by “disp_len” option of “3.3.7(1) udpopts”.) (Always displayed in JSON command.)

Example of command line usage

<Transmission completion message output when the transmission completion output setting (send_done) is turned ON by udpopts command

```
>udpopts send_done on
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1>
>
```

<Transmission completion message output when port number output setting (disp_port) is turned ON by udpopts command>

```
>udpopts send_done on
>udpopts disp_port on
>udpopts disp_len  off
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1> (3610)
>
```

<Transmission completion message output when transmission/reception data length output (disp_len) is turned ON by udpopts command>

```
>udpopts send_done on
>udpopts disp_len  on
>udpopts disp_port off
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1> (8)
>
```

<Transmission completion message output when the transmission port number setting is changed by udpopts command>

```
>udpopts send_done on
>udpopts disp_port on
>udpopts send_port 5000
>udpopts disp_len  on
>udps 2001:db8::1 0011223344556677
udpsd <2001:db8::1> (5000) (8)
>
```

Example of JSON command usage

<Transmission completion message output when the transmission completion output setting (send_done) is turned ON by udpopts command>

```
>{ "udpopts" : { "option" : "send_done", "enable" : true } }
>{ "udps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
{ "udpsd" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8 } }
>
```

<Transmission completion message output when the transmission port number setting is changed by udpopts command>

```
>{ "udpopts" : { "option" : "send_done", "enable" : true } }
>{ "udpopts" : { "option" : "send_port", "port" : 5000 } }
>{ "udps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
{ "udpsd" : { "ADDR" : "2001:db8::1", "port" : 5000, "len" : 8 } }
>
```

(2) tcpsd

Overview

The message outputs when completing the transmission of packets sent by “3.3.6(3) tcps” command.
This message is not displayed with the normal default settings.
Displayed when the display of the transmission completion message is enabled in the send_done option of "3.3.7(2) tcpopts".

Command-line display format

```
tcpsd <ADDR>[(port)] [(len)]
```

JSON command display format

```
{ "tcpsd" : { "ADDR" : <string>, "port" : <number>, "len" : <number> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	To display the destination IPv6 address.
port	number	1~65535	TCP-Server port number of TCP data transmission packet For the displayed port number, the TCP-Server port number used for the outgoing packet after TCP-Connection was established is displayed. (Refer to overview of "3.3.2(14) tcpcon" command for "TCP-Server address" / "TCP-Server port".) (Not normally displayed on command-line. It will be displayed when the display setting is enabled in the "disp_port" option of "3.3.7(2) tcpopts". (Always displayed in JSON command.)
len	number	1~65535	To display the transmission data length (Not normally displayed on command-line. It will be displayed when the display setting is enabled in the "disp_len" option of "3.3.7(2) tcpopts". (Always displayed in JSON command.)

Example of command line usage

<Transmission completion message output when the transmission completion output setting (send_done) is turned ON by tcpopts command>

```
>tcpopts send_done on
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1>
>
```

<Transmission completion message output when port number output setting (disp_port) is turned ON by tcpopts command>

```
>tcpopts send_done on
>tcpopts disp_port on
>tcpopts disp_len off
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (3610)
>
```

<Transmission completion message output when transmission/reception data length output (disp_len) is turned ON by tcpopts command>

```
>tcpopts send_done on
>tcpopts disp_len on
>tcpopts disp_port off
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (8)
>
```

<Transmission completion message output when port number/transmission data length output setting is turned ON by tcpopts command>

```
>tcpopts send_done on
>tcpopts disp_len on
>tcpopts disp_port on
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (3610) (8)
>
```

<Transmission completion message output when the transmission TCP-Server port number setting is changed by tcpopts command>

```
>tcpopts send_done on
>tcpopts disp_len on
>tcpopts disp_port on
>tcpopts send_port 5001
>tcps 2001:db8::1 0011223344556677
tcpsd <2001:db8::1> (5001) (8)
>
```

Example of JSON command usage

<Transmission completion message output when the transmission completion output setting (send_done) is turned ON by tcpopts command>

```
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 3610, "len" : 8 } }
>
```

<Transmission completion message output when the transmission TCP-Server port number setting is changed by tcpopts command>

```
>{ "tcpopts" : { "option" :  "send_done", "enable" : true } }
>{ "tcpopts" : { "option" :  "send_port", "port" : 5001 } }
>{ "tcps" : { "ADDR" : "2001:db8::1", "data" : "0011223344556677" } }
{ "tcpsd" : { "ADDR" : "2001:db8::1", "port" : 5001, "len" : 8 } }
>
```

4.3.3 Remote command message

(1) rmtmsg

Overview

This message displays the command execution result (command display content on the remote terminal) on the terminal that received the remote command from the terminal that issued the remote command by "3.3.9(1) rmtcmd" command
This message will not be displayed if a command that does not display the command execution result is sent by "3.3.9(1) rmtcmd" command.
Port number 48878 is used for UDP communication to send and receive messages by commands "3.3.9(1) rmtcmd" and "4.3.3(1) rmtmsg".

Command-line display format

```
rmtmsg <ADDR> <cmd>
```

JSON command display format

```
{ "rmtmsg" : { "ADDR" : "<string>", "cmd" : <JSON Object> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	IPv6 address transmission source (The terminal address that received the remote command and sent the execution result message by "3.3.9(1) rmtcmd" command.)
cmd	JSON object	command execution result message	Command statement to be executed on the remote terminal The command execution result message in command-line format: <command name> [argument 1] [argument 2] The command execution result message is according to the display format of each executed command Displayed contents of the command are output. The command execution result message in JSON format: {"<command name>": <argument 1 value>} {"<command name>": {"argument 1 name": <argument 1 value>,…}} For execution command statement in the JSON command, the command execution result message is displayed as the value of JSON Object for "cmd" which is the JSON format of rmtcmd mentioned above according to the JSON command display format of each command to be executed.

Example of command line usage

<To execute a single display format command (instance command) with rmtcmd (only one display item)>

```
>rmtcmd 2001:db8::2 instance
rmtmsg <2001:db8::2>: instance 0
>
```

<To execute a display format command (chan command) with multiple display items with rmtcmd>

```
>rmtcmd 2001:db8::2 chan
rmtmsg <2001:db8::2>: chan low=33<=>high=33,cur=33,base=920.600MHz num=1
>
```

<To execute a display format command (macf command) with multiple lines of display items with rmtcmd

```
>rmtcmd 2001:db8::2 macf
rmtmsg <2001:db8::2>: macf default             ( deny  )
rmtmsg <2001:db8::2>: macf <0001020304050601>   ( allow )
rmtmsg <2001:db8::2>: macf <0001020304050603>   ( allow )
>
```

Example of JSON command usage

<To execute a single display format command (instance command) with rmtcmd (only one display item)>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "instance" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "instance" : 0 } } }
>
```

<To execute a display format command (chan command) with multiple display items with rmtcmd>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "chan" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "chan" : { "low" : 33, "high" : 33, "cur" : 33, "base" :
 920600, "chnum" : 1 } } } }
>
```

<To execute a display format command (macf command) with multiple lines of display items with rmtcmd>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "macf" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "default" : "deny" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "<00010203040506
01>" } } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "macf" : { "role" : "allow", "MAC64B" : "<00010203040506
03>" } } } }
>
```

(2) rmtsd

Overview

This message outputs sent remote command message by "3.3.9(1) rmtcmd" command when the transmission is completed.
This message is not displayed with the normal default settings.
This message will display when the display of the transmission completion message is enabled with "send_done" option of "3.3.9(2) rmtopts".

Command-line display format

```
rmtsd <ADDR> <cmd>
```

JSON command display format

```
{ "rmterr" : { "ADDR" : <string>, "cmd" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
ADDR	string	<IPv6 address>	IPv6 address transmission source (The terminal address on which the remote command is sent by "3.3.9(1) rmtcmd" command.)
cmd	string	command execution result message	The command statement that sent the remote command by "3.3.9(1) rmtcmd" command.

* If the destination IPv6 address of rmtcmd is a multicast address, the confirmation of reception in the destination terminal will not be acknowledged and the error message mentioned above will not be displayed.

Example of command line usage

```
>rmtopts send_done off
>
>rmtcmd 2001:db8::2 mac
rmtmsg <2001:db8::2>: mac <0001020304050602>,LLA<fe80::201:203:405:602>
>
>rmtopts send_done on
>
>rmtcmd 2001:db8::2 mac
rmtsd <2001:db8::2> mac
rmtmsg <2001:db8::2>: mac <0001020304050602>,LLA<fe80::201:203:405:602>
>
```

Example of JSON command usage

```
>{ "rmtopts" : { "option" : "send_done", "enable" : false } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "mac" : null } } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "mac" : { "MAC64B" : "0001020304050602", "IP128BLLA" : "
fe80::201:203:405:602" } } } }
>
>{ "rmtopts" : { "option" : "send_done", "enable" : true } }
>
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "mac" : null } } }
{ "rmtsd" : { "ADDR" : "2001:db8::2", "cmd" : "{ \"mac\" : null }" } }
{ "rmtmsg" : { "ADDR" : "2001:db8::2", "cmd" : { "mac" : { "MAC64B" : "0001020304050602", "IP128BLLA" : "
fe80::201:203:405:602" } } } }
>
```

(3) rmterr

Overview

The transmission error message of the remote command is displayed at the terminal that sent the remote command by "3.3.9(1) rmtcmd" command.
Error messages related to the command execution result on the terminal that received the remote command are displayed by "4.3.3(1) rmtmsg".

Command-line display format

```
rmterr <desc> <ADDR> <cmd>
```

JSON command display format

```
{ "rmterr" : { "desc" : <string>, "ADDR" : <string>, "cmd" : <string> } }
```

Display format contents

Argument Name	JSON Argument Type	Display Value	Description
desc	string	error overview	Error overview
ADDR	string	<IPv6 address>	IPv6 address destination (Terminal address that sent the remote command by "3.3.9(1) rmtcmd" command.)
cmd	string	command execution result message	The command statement that sent the remote command by "3.3.9(1) rmtcmd" command.

* If the destination IPv6 address of rmtcmd is a multicast address, the confirmation reception of the destination terminal is not acknowledged and the above error message will not be displayed**.**

Example of command line usage

<When the executed rmtcmd cannot be confirmed on the destination terminal (when rmtcmd has not arrived)>

```
>rmtcmd 2001:db8::2 macf allow 0001020304050605
rmterr unreached message <2001:db8::2> macf allow 0001020304050605
>
```

Example of JSON command usage

<When the executed rmtcmd cannot be confirmed on the destination terminal (when rmtcmd has not arrived)>

```
>{ "rmtcmd" : { "ADDR" : "2001:db8::2",  "cmd" : { "macf" : { "role" : "allow",  "MAC64B" : "000102030405
0605" } } } }
{ "rmterr" : { "desc" : "unreached message", "ADDR" : "<2001:db8::2>", "cmd" : "{ \"macf\" : { \"role\" :
 \"allow\", \"MAC64B\" : \"0001020304050605\" } }" } }
>
```

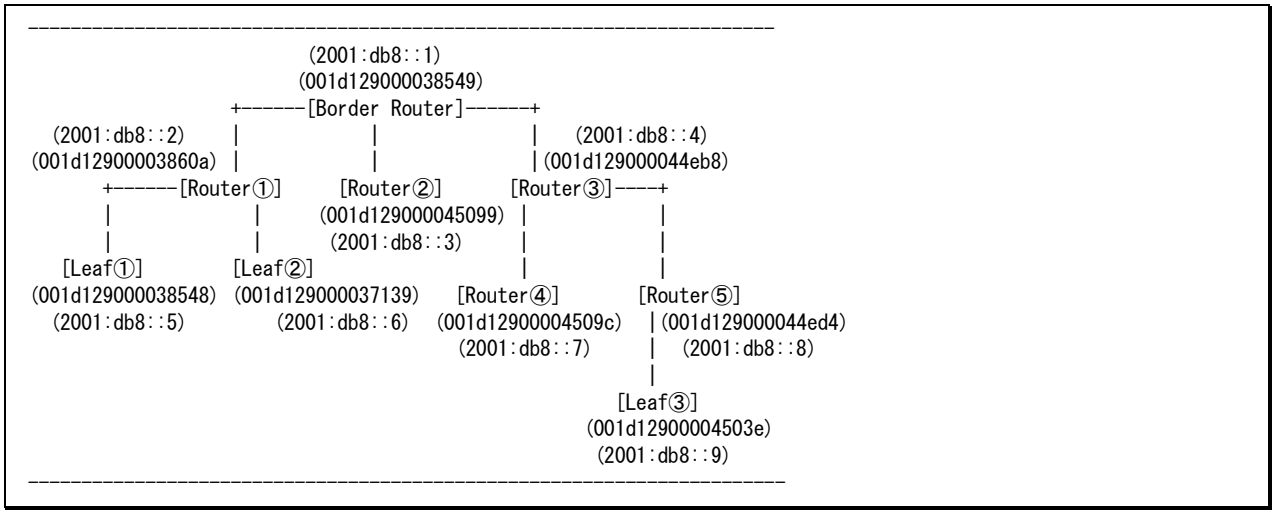
5. Network Topology Construction Method

This section is to describe how to build a network topology.

5.1 Multi-hop network topology

5.1.1 Configuration

Confirm that the packet is forwarded by multi-hop under the following configuration.



The MAC address of each connected node and the assigned IPv6 address are as follows.

No	Node name	MAC address	IPv6 address
1	Border Router	001d129000038549	2001:db8::1
2	Router ①	001d12900003860a	2001:db8::2
3	Router ②	001d129000045099	2001:db8::3
4	Router ③	001d129000044eb8	2001:db8::4
5	Leaf ①	001d129000038548	2001:db8::5
6	Leaf ②	001d129000037139	2001:db8::6
7	Router ④	001d12900004509c	2001:db8::7
8	Router ⑤	001d129000044ed4	2001:db8::8
9	Leaf ③	001d12900004503e	2001:db8::9

5.1.2 Setting

Parameter setting of each terminal is performed. Parameter settings are stored in the non-volatile area by save command. After save command execution, it will operate based on set value after rebooting. (Setting will not be required next time.)
* Please be warned that parameters will be deleted if clear command or firmware update is performed.
* For macf command, network topology mentioned above is described as being built within the same radio wave range (such as on a desk).
(Setting of macf command is not required in the actual network field.)
* The settings are just an example. Perform the settings according to one's environment.

(1) Setting of Border Router

```
clear

(Flash-wait by clearcommand)

reset
(Reboot by reset command)

macf deny
macf allow 001d12900003860a
macf allow 001d129000045099
macf allow 001d129000044eb8

chan 33 59

ip 2001:db8::1/48

leaseip 001d12900003860a 2001:db8::2
leaseip 001d129000045099 2001:db8::3
leaseip 001d129000044eb8 2001:db8::4
leaseip 001d129000038548 2001:db8::5
leaseip 001d129000037139 2001:db8::6
leaseip 001d12900004509c 2001:db8::7
leaseip 001d129000044ed4 2001:db8::8
leaseip 001d12900004503e 2001:db8::9

atstart 1
save

(Flash-wait by save command)

reset
```

(2) Setting of Router①

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000038549
macf allow 001d129000045099
macf allow 001d129000044eb8
macf allow 001d129000038548
macf allow 001d129000037139

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(3) Setting of Router②

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000038549
macf allow 001d12900003860a
macf allow 001d129000044eb8

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(4) Setting of Router③

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000038549
macf allow 001d12900003860a
macf allow 001d129000045099
macf allow 001d12900004509c
macf allow 001d129000044ed4

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(5) Setting of Leaf①

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d12900003860a
macf allow 001d129000037139

chan 33 59

atstart 3
save

(Flash-wait by save command)

reset
```

(6) Setting of Leaf②

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d12900003860a
macf allow 001d129000038548

chan 33 59

atstart 3
save

(Flash-wait by save command)

reset
```

(7) Setting of Router④

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000044eb8
macf allow 001d129000044ed4
macf allow 001d12900004503e

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(8) Setting of Router⑤

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000044eb8
macf allow 001d12900004509c
macf allow 001d12900004503e

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(9) Setting of Leaf③

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d12900004509c
macf allow 001d129000044ed4

chan 33 59

atstart 3
save

(Flash-wait by save command)

reset
```

5.1.3 Startup

This is to describe the startup sequence of each node after executing the setting.
Beforehand, power supply of all Wi-SUN modules must be turned off.

(1) Startup sequence

① Turn on Border Router.
② Turn on Router①, Router② and Router③.
③ Turn on Leaf①, Leaf②, Router④ and Router⑤.
④ Turn on Leaf③.

(2) Source routing registration confirmation method

Source routing information to Border Router is collected after a few minutes to tens of minutes after executing according to the startup sequence.

Packet transmission / reception by multi-hop cannot be performed unless the source routing information is collected.

(a) Border Router terminal source routing confirmation method

(The following are the states where the routing information of the node is collected on all Border Router terminals.)

```
>rplsr
- Routing links (8 in total):
-- 2001:db8::2 to 2001:db8::1(lifetime: 7156 seconds)
-- 2001:db8::3 to 2001:db8::1(lifetime: 7164 seconds)
-- 2001:db8::4 to 2001:db8::1(lifetime: 7163 seconds)
-- 2001:db8::7 to 2001:db8::4 (lifetime: 7152 seconds)
-- 2001:db8::8 to 2001:db8::4 (lifetime: 7200 seconds)
-- 2001:db8::5 to 2001:db8::2 (lifetime: 7183 seconds)
-- 2001:db8::6 to 2001:db8::2 (lifetime: 7172 seconds)
-- 2001:db8::9 to 2001:db8::7 (lifetime: 7156 seconds)
>
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 128
rplinf my DAGRANK  : 1
rplinf parent      : MAC<NONE>,LLA<NONE>,GBL<NONE>
>
```

(b) Router / Leaf terminal source routing confirmation method

(The following state is the state where the routing information to the parent node is set on all Router / Leaf terminals.)

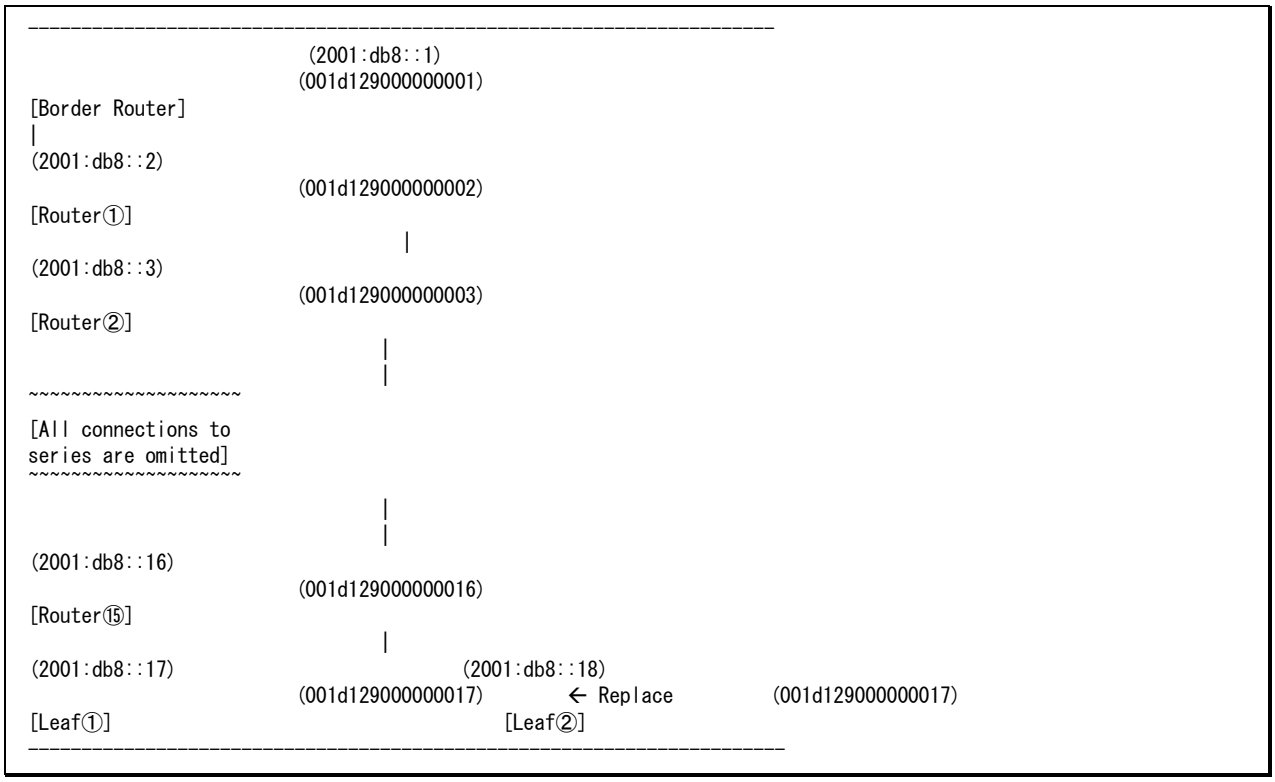
```
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 341
rplinf my DAGRANK  : 2
rplinf parent      : MAC<0001020304050601>,LLA<fe80::201:203:405:601>,GBL<2001:db8::1>
>
```

5.2 Terminal replacement (replacement)

This chapter describes the command setting procedure when replacing a failed terminal connected to the network with a new terminal. Normally, replacement is possible even without this procedure. However, this is necessary when the maximum number of authenticated / connected terminal information managed by the Border Router is exceeded. The number of terminal information that can be managed by Border Router depends on the platform. (16 by default)

5.2.1 Configuration

Confirm that the network configuration and terminal replacement can be performed with the following configuration.



The MAC address of each connected node and the assigned IPv6 address are as follows.

No	Node name	MAC address	IPv6 address
1	Border Router	001d129000000001	2001:db8::1
2	Router①	001d129000000002	2001:db8::2
3	Router②	001d129000000003	2001:db8::3
4	Router③	001d129000000004	2001:db8::4
5	Router④	001d129000000005	2001:db8::5
6	Router⑤	001d129000000006	2001:db8::6
7	Router⑥	001d129000000007	2001:db8::7
8	Router⑦	001d129000000008	2001:db8::8
9	Router⑧	001d129000000009	2001:db8::9
10	Router⑨	001d129000000010	2001:db8::10
11	Router⑩	001d129000000011	2001:db8::11
12	Router⑪	001d129000000012	2001:db8::12
13	Router⑫	001d129000000013	2001:db8::13
14	Router⑬	001d129000000014	2001:db8::14
15	Router⑭	001d129000000015	2001:db8::15
16	Router⑮	001d129000000016	2001:db8::16
17	Leaf①	001d129000000017	2001:db8::17
18	Leaf②	001d129000000018	2001:db8::18

5.2.2 Setting

Parameter setting of each terminal is performed. Parameter settings are stored in the non-volatile area by save command.
After save command execution, it will operate based on set value after rebooting. (Setting is not required next time.)
* Please be warned that parameters will be deleted if clear command or firmware update is performed.
(Setting of macf command is not required in the actual network field.) * For the macf command, the network topology mentioned above is described as being built within the same radio wave range (such as on a desk).
* The settings are just an example. Perform the settings according to one's environment.

(1) Setting of Border Router

```
clear

(Flash-wait by clearcommand)

reset
(Reboot by reset)

macf deny
macf allow 001d129000000002  ← Terminal MAC address of Router①

chan 33 59

ip 2001:db8::1/48

leaseip 001d129000000002 2001:db8::2
leaseip 001d129000000003 2001:db8::3
leaseip 001d129000000004 2001:db8::4
leaseip 001d129000000005 2001:db8::5
leaseip 001d129000000006 2001:db8::6
leaseip 001d129000000007 2001:db8::7
leaseip 001d129000000008 2001:db8::8
leaseip 001d129000000009 2001:db8::9
leaseip 001d129000000010 2001:db8::10
leaseip 001d129000000011 2001:db8::11
leaseip 001d129000000012 2001:db8::12
leaseip 001d129000000013 2001:db8::13
leaseip 001d129000000014 2001:db8::14
leaseip 001d129000000015 2001:db8::15
leaseip 001d129000000016 2001:db8::16
leaseip 001d129000000017 2001:db8::17


atstart 1
save

(Flash-wait by save command)

reset
```

(2) Setting of Router①

```
clear

(Flash-wait by clear command)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000000001  ← Terminal MAC address of Border Router
macf allow 001d129000000003  ← Terminal MAC address of Router②

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(3) Setting of Router②

```
clear

(Flash-wait by clear command)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000000002  ← Termial MAC address of Border Router
macf allow 001d129000000004  ← Termial MAC address of Router②

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(4) Setting of Router③~Router⑭

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d1290000000xx  ← The MAC address of the terminal one level above itself
macf allow 001d1290000000xx  ← The MAC address of the terminal one level below itself

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(5) Setting Router⑮

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000000015  ← Terminal MAC address of Router⑭
macf allow 001d1290000000xx  ← Terminal MAC address of Leaf①

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

(6) Setting of Leaf①~Leaf②

```
clear

(Flash-wait by clearcommand)

reset

(Reboot by reset command)

macf deny
macf allow 001d129000000016  ← Terminal MAC address of Router⑮

chan 33 59

atstart 2
save

(Flash-wait by save command)

reset
```

5.2.3 Startup

This is to describe the startup sequence of each node after executing the setting.
Beforehand, power supply of all Wi-SUN modules must be turned off.

(1) Startup sequence

① Turn on Border Router.
② Turn on the power one by one starting from the Router that is closest to the Border Router.
③ Turn on Leaf①.

(2) Source routing registration confirmation method

Source routing information is collected in the Border Router within a few minutes to tens of minutes after execution according to the startup sequence.
Packet transmission / reception by multi-hop cannot be performed unless the source routing information is collected.

(a) Border Router terminal source routing confirmation method
(The following are the states where the routing information of all nodes is collected on the Border Router terminal.)

```
>rplsr
- Routing links (16 in total):
-- 2001:db8::2 to 2001:db8::1(lifetime: 7156 seconds)
-- 2001:db8::3 to 2001:db8::2 (lifetime: 7164 seconds)
-- 2001:db8::4 to 2001:db8::3 (lifetime: 7163 seconds)
-- 2001:db8::5 to 2001:db8::4 (lifetime: 7152 seconds)
-- 2001:db8::6 to 2001:db8::5 (lifetime: 7200 seconds)
-- 2001:db8::7 to 2001:db8::6 (lifetime: 7183 seconds)
-- 2001:db8::8 to 2001:db8::7 (lifetime: 7172 seconds)
-- 2001:db8::9 to 2001:db8::8 (lifetime: 7156 seconds)
-- 2001:db8::10 to 2001:db8::9 (lifetime: 7152 seconds)
-- 2001:db8::11to 2001:db8::10 (lifetime: 7200 seconds)
-- 2001:db8::12 to 2001:db8::11(lifetime: 7183 seconds)
-- 2001:db8::13 to 2001:db8::12 (lifetime: 7172 seconds)
-- 2001:db8::14 to 2001:db8::13 (lifetime: 7156 seconds)
-- 2001:db8::15 to 2001:db8::14 (lifetime: 7183 seconds)
-- 2001:db8::16 to 2001:db8::15 (lifetime: 7172 seconds)
-- 2001:db8::17 to 2001:db8::16 (lifetime: 7172 seconds)
>
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 128
rplinf my DAGRANK  : 1
rplinf parent      : MAC<NONE>,LLA<NONE>,GBL<NONE>
>
```

(b) Router / Leaf terminal source routing confirmation method
(The following are states where the routing information to the parent node is set on all Router / Leaf terminals.)

```
>rplinf
rplinf DODAG ID    : <2001:db8::1>
rplinf instance ID : 0
rplinf my rank     : 341
rplinf my DAGRANK  : 2
rplinf parent      : MAC<0001020304050601>,LLA<fe80::201:203:405:601>,GBL<2001:db8::1>
>
```

(c) Authentication / connected node confirmation method

(The following are states where all nodes are authenticated and connected on the Border Router terminal.)

```
>fnode
fnode <001d129000000002>,<2001:db8::2>
fnode <001d129000000003>,<2001:db8::3>
fnode <001d129000000004>,<2001:db8::4>
fnode <001d129000000005>,<2001:db8::5>
fnode <001d129000000006>,<2001:db8::6>
fnode <001d129000000007>,<2001:db8::7>
fnode <001d129000000008>,<2001:db8::8>
fnode <001d129000000009>,<2001:db8::9>
fnode <001d129000000010>,<2001:db8::10>
fnode <001d129000000011>,<2001:db8::11>
fnode <001d129000000012>,<2001:db8::12>
fnode <001d129000000013>,<2001:db8::13>
fnode <001d129000000014>,<2001:db8::14>
fnode <001d129000000015>,<2001:db8::15>
fnode <001d129000000016>,<2001:db8::16>
fnode <001d129000000017>,<2001:db8::17>
>
```

5.2.4 Terminal replacement (replacement)

Replaces Leaf① and Leaf② as terminal replacement.

(1) Replacement work

① Turn off the power of the replacement target terminal (Leaf①) involved in the network.
② Change the mac address filter on the upper terminal (Router⑮) at the position involved in the network.

```
>macf
macf allow 001d129000000015 ← Terminal mac address of Router ⑮
macf allow 001d129000000017 ← Terminal mac address of Leaf ①

>macf allow 001d129000000018 ← Allow the terminal mac address of Leaf②
>macf del 001d129000000017 ← Delete the terminal mac address setting of Leaf ① (no need to do)
```

③ Delete the terminal information of Leaf① on the Border Router.

```
>fnode del 001d129000000017
inf 4c,3b,72e8,120  {    RPL: deleted node <2001:db8::17>
leaseip del 001d129000000017
>
```

④ Confirm that the terminal information of Leaf① has been deleted on the Border Router.

```
>fnode
fnode <001d129000000002>,<2001:db8::2>
fnode <001d129000000003>,<2001:db8::3>
fnode <001d129000000004>,<2001:db8::4>
fnode <001d129000000005>,<2001:db8::5>
fnode <001d129000000006>,<2001:db8::6>
fnode <001d129000000007>,<2001:db8::7>
fnode <001d129000000008>,<2001:db8::8>
fnode <001d129000000009>,<2001:db8::9>
fnode <001d129000000010>,<2001:db8::10>
fnode <001d129000000011>,<2001:db8::11>
fnode <001d129000000012>,<2001:db8::12>
fnode <001d129000000013>,<2001:db8::13>
fnode <001d129000000014>,<2001:db8::14>
fnode <001d129000000015>,<2001:db8::15>
fnode <001d129000000016>,<2001:db8::16>
>
```

⑤ Set the IPv6 address of Leaf② on the Border Router.

```
>
> leaseip 001d129000000018 2001:db8::18
>
```

⑥ Turn on the power of Leaf②.

(2) Confirmation of terminal replacement

Joins the network in a few minutes after starting Leaf②.
When unable to join the network, confirmation of source routing information and authentication / connection information is not possible

(a) Border Router Terminal source routing confirmation method
(Leaf② terminal is displayed at the end in the rplsr command on the Border Router terminal to which the Leaf② terminal is connected)

```
>rplsr
- Routing links (16 in total):
-- 2001:db8::2 to 2001:db8::1(lifetime: 7156 seconds)
-- 2001:db8::3 to 2001:db8::2 (lifetime: 7164 seconds)
-- 2001:db8::4 to 2001:db8::3 (lifetime: 7163 seconds)
-- 2001:db8::5 to 2001:db8::4 (lifetime: 7152 seconds)
-- 2001:db8::6 to 2001:db8::5 (lifetime: 7200 seconds)
-- 2001:db8::7 to 2001:db8::6 (lifetime: 7183 seconds)
-- 2001:db8::8 to 2001:db8::7 (lifetime: 7172 seconds)
-- 2001:db8::9 to 2001:db8::8 (lifetime: 7156 seconds)
-- 2001:db8::10 to 2001:db8::9 (lifetime: 7152 seconds)
-- 2001:db8::11to 2001:db8::10 (lifetime: 7200 seconds)
-- 2001:db8::12 to 2001:db8::11(lifetime: 7183 seconds)
-- 2001:db8::13 to 2001:db8::12 (lifetime: 7172 seconds)
-- 2001:db8::14 to 2001:db8::13 (lifetime: 7156 seconds)
-- 2001:db8::15 to 2001:db8::14 (lifetime: 7183 seconds)
-- 2001:db8::16 to 2001:db8::15 (lifetime: 7172 seconds)
-- 2001:db8::18 to 2001:db8::16 (lifetime: 7172 seconds)
>
```

(b) Authentication / connected node information
(Leaf② terminal is displayed at the end in the fnode command on the Border Router terminal to which the Leaf② terminal is connected)

```
>fnode
fnode <001d129000000002>,<2001:db8::2>
fnode <001d129000000003>,<2001:db8::3>
fnode <001d129000000004>,<2001:db8::4>
fnode <001d129000000005>,<2001:db8::5>
fnode <001d129000000006>,<2001:db8::6>
fnode <001d129000000007>,<2001:db8::7>
fnode <001d129000000008>,<2001:db8::8>
fnode <001d129000000009>,<2001:db8::9>
fnode <001d129000000010>,<2001:db8::10>
fnode <001d129000000011>,<2001:db8::11>
fnode <001d129000000012>,<2001:db8::12>
fnode <001d129000000013>,<2001:db8::13>
fnode <001d129000000014>,<2001:db8::14>
fnode <001d129000000015>,<2001:db8::15>
fnode <001d129000000016>,<2001:db8::16>
fnode <001d129000000018>,<2001:db8::18>
>
```