SmartMesh IP VManager CLI Guide

Table of Contents

1 About This Guide

1.1 Related Documents

The following documents are available for the SmartMesh IP network:

1.1.1 Getting Started with a Starter Kit

- SmartMesh VManager Easy Start Guide - walks you through basic VManager installation and a few tests to make sure your network is working.
- SmartMesh IP Embedded Manager Easy Start Guide - walks you through basic embedded manager installation and a few tests to make sure your network is working.
- SmartMesh IP Embedded Manager Tools Guide - the installation section contains instructions for installing the serial drivers and example programs used in the Easy Start Guide and other tutorials.

1.1.2 User's Guide

- SmartMesh IP User's Guide - describes network concepts, and discusses how to drive mote and manager APIs to perform specific tasks, e.g. to send data or collect statistics. This document provides context for the API guides. It also contains a glossary of SmartMesh terms.

1.1.3 Interfaces for Interaction with a Device

There are two interfaces for interaction with a Manager - an Application Programming Interface (API) for programmatic interaction, and a Command Line Interface (CLI) for human interaction.

- SmartMesh IP Embedded Manager CLI Guide - used for human interaction with an embedded manager (e.g. during development of a client, or for troubleshooting). This document covers connecting to the CLI and its command set.
- SmartMesh IP Embedded Manager API Guide - used for programmatic interaction with an embedded manager. This document covers connecting to the API and its command set.
- SmartMesh IP VManager CLI Guide - used for human interaction with a VManager (e.g. during development of a client, or for troubleshooting). This document covers connecting to the CLI and its command set.
- SmartMesh IP VManager API Guide - used for programmatic interaction with a VManager. This document covers connecting to the API and its command set.
- SmartMesh IP Mote CLI Guide - used for human interaction with a mote (e.g. during development of a sensor application, or for troubleshooting). This document covers connecting to the CLI and its command set.

- SmartMesh IP Mote API Guide - used for programmatic interaction with a mote. This document covers connecting to the API and its command set.

1.1.4 Access Point Motes

- SmartMesh IP User's Guide - describes reprogramming DC2274 for use as an Access Point Mote.
- VManager AP Bridge User's Guide - user's guide for the Access Point Bridge reference software

1.1.5 Software Development Tools

- Dustcloud.org - contains documentation and links to various open source software tools for iexercising mote and manager APIs and visualizing the network.

1.1.6 Application Notes

- SmartMesh IP Application Notes - Cover a wide range of topics specific to SmartMesh IP networks and topics that apply to SmartMesh networks in general.

1.1.7 Documents Useful When Starting a New Design

- The Datasheet for the mote being used, e.g. the LTC5800-IPM SoC, or one of the modules based on it.
- The Datasheet for the embedded manager being used, e.g. the LTC5800-IPR SoC, or one of the embedded managers based on it.
- A Hardware Integration Guide for the mote/manager SoC or module - this discusses best practices for integrating the SoC or module into your design.
- A Hardware Integration Guide for the embedded manager - this discusses best practices for integrating the embedded manager into your design.
- A Board Specific Integration Guide - For SoC motes and Managers. Discusses how to set default IO configuration and crystal calibration information via a "fuse table".
- Hardware Integration Application Notes - contains an SoC design checklist, antenna selection guide, etc.
- The ESP Programmer Guide - a guide to the DC9010 Programmer Board and ESP software used to load firmware on a device.

1.1.8 Software

- ESP software - used to program firmware images onto a mote or module. Described in the ESP Programmer Guide.
- Fuse Table software - used to construct the fuse table as discussed in the Board Specific Configuration Guide.

1.1.9 Other Useful Documents

- A list of Frequently Asked Questions.

1.2 Conventions Used

The following conventions are used in this document:

`Computer type` indicates information that you enter, such as specifying a URL.

Bold type indicates buttons, fields, menu commands, and device states and modes.

Italic type is used to introduce a new term, and to refer to APIs and their parameters.

Tips provide useful information about the product.

Informational text provides additional information for background and context

Notes provide more detailed information about concepts.

Warning! Warnings advise you about actions that may cause loss of data, physical harm to the hardware or your person.

```
code blocks display examples of code
```

The CLI commands are described using the following notations and terminology:

|	Indicates alternatives for a field. For example, *<macAddr>* | *<moteId>* indicates that you can specify a mote by its mote ID or MAC address.
< >	Indicates a required field.
{ }	Indicates a group of fields.
[ ]	Indicates an optional field.

MAC address	When specifying a MAC address, do not use spaces. You may omit hyphens (though this document uses them to enhance readability).
	
	The following examples are all valid:
	
	`$> show mote 00000000000022CA`
	
	`$> show mote 00-00-00-00-00-00-22-CA`

1.3 Revision History

Revision	Date	Description
1	12/15/2015	Initial Release
2	08/19/2016	Phase I Production
3	11/08/2016	Added show apcs command; Updated show mote/ap command; Numerous minor improvements

2 Introduction

This guide describes the commands that you can send to a SmartMesh IP VManager by logging on to its Command Line Interface (CLI). The CLI is available via ssh connection to the Manager server. The CLI is intended for human interaction with a manager, e.g. during development, or for interactive troubleshooting. Most commands are atomic - a command and its arguments are typed into the CLI, and a response is returned. For example, the `help` command returns a list of possible commands. Commands that perform network operations are not atomic - they generate output asynchronously after receiving a response. Traces are not atomic - once started, they generate output asynchronously until cancelled.

For a machine-to-machine communications (e.g. a host program talking to the manager), the Application Programming Interface (API) is used. See the SmartMesh IP VManager API Guide for details.

2.1 Manager Connection

To access the CLI, you must first connect to the Manager Linux shell. This can be done using any terminal program that supports ssh, such as PuTTY or TeraTerm on Windows, or from any console/terminal window in Linux/OS X. You must know the IP address of the Manager server to ssh - e.g. if the server is at 192.168.1.100, then you would use:

```
$ ssh dust@192.168.1.100
```

You will be prompted for a password. For the user **dust**, the default password is also **dust**. For the SSH connection, users are managed through the system's user management commands.

Issue the `console` command from the Linux shell to start the CLI. The default username is **dust** and password is **dust**. For the Console and API connections, users are managed internally within the VManager configuration.

```
dust@voyager-vm:~$ console
Welcome to the Voyager CLI Console on Linux
Version 0.1.0.10

Enter your username: dust
dust's password:
$>
```

Note that although the default username and password (used to log into the console) are the same as the Linux login, these are separate logins and can be re-configured separately.

2.1.1 Managing User Accounts

Once logged into CLI, the user can change the password (or any other user field) for the default user account using the `config set user` CLI command:

```
$> show user dust
User information:
   Id:        dust
   Privilege: USER
   Description: duster

$> config set user dust description="a dust user"
Done

$> config reload users
Done

$> show user dust
User information:
   Id:        dust
   Privilege: USER
   Description: a dust user
```

Note that once the persistent user database has been modified, it must be reloaded to take effect.Similarly, new users can be created, and unneeded users can be deleted:

```
$> show users
User configuration:
Id:     dust

$> config set user bobo description="bobo" password="free4willy" privilege=user
Done

$> config reload users
Done

$> show users
User configuration:
Id:     dust
Id:     bobo

$> config delete user bobo
Done

$> config reload users
Done

$> show users
User configuration:
Id:     dust
```

2.1.2 Helpful tips

This manual describes the console commands available in the SmartMesh IP VManager. The console is case sensitive.

Several console shortcuts are available for command entry:

- The **up-arrow** key cycles through a history of previous commands
- The **tab** key autocompletes partially entered command names
- The **#** prefix can be used to simplify MAC addresses, e.g. `show mote #38-01-02` expands to `show mote 00-17-0D-00-00-38-01-02`

Commands that generate multi-page output can be invoked with a `-p` (or `--page`) switch to produce paginated output - pressing the **return** key advances to the next page.

2.1.3 Stored Versus Active Configuration

VManager distinguishes between active configuration parameters from those stored for use the next time the manager is started.

- The `show` commands display the active configuration
- The `config set/get` commands store or display stored parameters
- The `config reload` command overwrites the active configuration for a particular module with the stored version
- The `config restore` command overwrites the stored configuration with default values, aka "factory restore"

2.1.4 API Equivalents

API equivalents to CLI commands are given in Swagger format, e.g.:

```
POST /path/to/a/resource
```

See the Swagger documentation for JSON format of requests and responses for the equivalent API call.

3 Commands

3.1 clear

Description

This command clears the console screen

Syntax

```
clear
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> clear
```

3.2 config

The config commands interact with the stored inactive configuration settings. To see the currently used active settings, use the *show* command.

3.2.1 config delete

Description

Delete an item from a list in stored configuration.

Syntax

```
config delete acl <macAddr> |
              dcl <macAddr> |
              user <userId>
```

Parameters

Parameter (one of the following)	Description
acl	Delete a mote from the access control list, *macAddr, joinkey*
dcl	Delete a mote from the deny control list, *macAddr*
user	Delete a user from the system

Example

```
$>  config delete dcl 00-17-0D-00-00-12-34-56

Done

$>  config delete user mrdusty

Done
```

3.2.2 config deletei

Description

Deletes the contents of an INI value that was changed by a user, and resets it to the default value. This does not delete the actual parameter, its value is reset to the system defaults. If a non-existent parameter is entered, no error message is returned.

> ⚠ This command requires superuser privileges - see the su command. INI parameters should only be changed under Linear guidance to achieve specific performance goals.

Syntax

```
config deletei <param>
```

Parameters

Parameter	Description
param	Parameter to be reset

Example

```
#> config deletei BWMULT
Done
```

3.2.3 config get

Description

Display stored configuration for the requested item(s). When the -p or --page option is used, output is paused after each page.

Syntax

```
config get [-p|--page] acl <macAddr> |
                       dcl <macAddr> |
                       network |
                       system |
                       user <userId> |
                       users
```

Parameters

Parameter (one of the following)	Description
acl	Get the stored access control list (acl) configuration - entire acl or entry for mote *macAddr*
dcl	Get the stored deny control list (dcl) configuration - entire dcl or entry for mote *macAddr*
network	Get the stored network configuration
system	Get the stored system configuration
user	Get the stored user configuration for user *userId*
users	Get the stored users configuration

Example

```
$>  config get -p acl
00-17-0D-00-00-12-34-AA
00-17-0D-00-00-12-34-AB

$>  config get dcl
00-17-0D-00-00-12-34-56

$> config get network
Network configuration:
   networkId:              294
   topologyType:           MESH
   ccaMode:                False
   dsFrameMultiplier:      1
   basePkPeriod:           15000
   dsFrameSize:            512
   ipAddrMask:             FFFF:FFFF:FFFF:FFFF::
   commonJoinKey:          DUSTNETWORKSROCK
   joinSecurityType:       COMMON_SKEY
   minServicePkPeriod:     100
   maxMotes:               2000
   numParents:             2
   ipAddrPrefix:           FE80::
   dsFrameMultiplierDelay: 3600000
   usFrameSize:            1024
   channelList:            32767

$> config get system
System configuration:
   Name:
   Location:
   CLI timeout: 0 minutes

$> config get users
User configuration:
Id:     dust

$> config get user dust
User information:
   Id:        dust
   Password:  ****
   Privilege: USER
   Description: duster
```

3.2.4 config geti

Description

Displays internal INI settings. If a parameter *param* isn't specified, all parameters are listed. An asterisk (*) can be used as a wildcard in the *param* field.

> ⚠ This command requires superuser privileges - see the su command. There are a large number of tuning parameters (not documented here) that have been optimized for general use - most use cases will use the default parameters. INI parameters should only be changed under Linear guidance to achieve specific performance goals.

Syntax

```
config geti [param]
```

Parameters

Parameter	Description
param	Name of the INI parameter

Example

```
#> config geti *
MNGRINI Module configuration :
ADV_SEND_TO:180000
ADV_TIME_DELAY:3600000
ADV_TIME_OFF:1000
ADV_TIME_ON:1000
  .....
#> config geti NUMBCAST
MNGRINI Module configuration :
NUMBCAST:2
```

3.2.5 config set

Description

Change the stored inactive configuration settings. If an item does not exist, it will be created if appropriate. Fields arguments whose names contains spaces, need to be wrapped in quotes. For example sysName = "Foo Bar"

All settings changed in inactive memory have no effect on the system until they are loaded into active memory. To do so, use the `config reload <module> | all` command. Also note that all settings marked with an asterisk (*) require a system reset before they can take effect, using the `reset --reload` command.

Syntax

```
config set acl <macAddr> joinKey=<joinKey> |
           dcl <macAddr> |
           network {<field=val> [field2=val2] ...} |
           system {<field=val> [field2=val2] ...} |
           user <userId> {<field=val> [field2=val2] ...}
```

Parameters

Parameter (one of the following)	Description
acl	Store the access control list entry tuple *macAddr, joinkey,* where *joinkey* is a 16-byte key
dcl	Store the deny control list entry for mote *macAddr*
network	Store a network configuration field
system	Store a system configuration field
user	Store a user's configuration (Note: UserID can ONLY small letters and numbers are valid)

Example

```
$>  config set acl 00-17-0D-00-00-60-04-B0 joinkey=445553544E4554574F524B53524F434B
Done

$>  config set dcl 00-17-0D-00-00-12-34-56
Done

$>  config set system sysName=”Foo” location=”Bar Baz”
Done

$>  config set user mrdusty privilege=user
Done

$>  config reload all
Done
```

config set network parameters

The following network parameters are settable. Note that all items marked with an asterisk (*) require a *reset network --reload* command to take effect. This will cause a full network reset :

Parameter	Description	Default Value
networkId *	Network ID	1229
topologyType *	Routing and cascading (MESH), routing and non-cascading (EVENT), or non-routing (STAR)	MESH
downFrameMultiplierDelay *	Time from first mote join until switching to use the downstream frame multiplier (ms)	3600000
ccaMode *	Mode used for clear channel assessment. 0=False=off, 1=True=on	False
ipAddrPrefix *	IP Address Prefix for WSN	FE80::
basePkPeriod	Minimum bandwidth to allocate to each device (ms)	15000
downFrameMultiplier *	Downstream frame multiplier for steady state. The downstream frame is extended from *downFrameSize* to *downFrameSize*downFrameMultiplier* after the network has formed	1
joinSecurityType *	Security mode for joining devices - one of COMMON_SKEY, COMMON_SKEY_QUARANTINE, UNIQUE_SKEY)	COMMON_SKEY
commonJoinKey	Common join key (only valid if joinSecurityMode is set to COMMON_SKEY or COMMON_SKEY_QUARANTINE. Must be in base64 format and be 128 bits long. For security reasons, the key is not returned in GET response	44 55 53 54 4E 45 54 57 4F 52 4B 53 52 4F 43 4B
minServicePkPeriod	Minimum (fastest) service request allowed (ms)	100
downFrameSize *	Downstream frame size (slots)	512
numParents	Desired number of parents for each mote	2
channelList *	Bitmap of used channels (b0 = IEEE channel 11)	32767
upFrameSize *	Upstream frame size (slots)	1024
gpsMode *	Clock source for the network, GPS or free-running. 0=False=off, 1=True=on	False

Example

```
$> config set network networkId=1229
Done
$> reset network --reload
```

config set system parameters

The following system parameters are settable:

Parameter	Description
sysName	Name string assigned by the user
location	Location string assigned by the user
cliTimeout	CLI timeout for inactive session (s)

Note that strings containing spaces must be entered in quotes, e.g. "Mr Dusty".

config set user parameters

Create a new user. The following user parameters are settable:

Parameter	Description
userId	User id (string), see userId limitations below
description	Description of the user
password	User password
privilege	User privilege level, *user* or *viewer*

There are two possible privilege settings. The *user* privilege has access to all (non-su) commands and can change system settings, while the *viewer* privilege can only view settings.

The userId must consist of only lowercase letters, numbers, and the _ and - characters, and must begin with a letter. No uppercase letters, spaces, or other characters may be part of the userId.

Example

```
$>  config set user mrdusty privilege=user password=mypw
Done
```

3.2.6 config seti

Description

Change internal INI parameters.

> ⚠ This command requires superuser privileges - see the su command. There are a large number of tuning parameters (not documented here) that have been optimized for general use - most use cases will use the default parameters. INI parameters should only be changed under Linear guidance to achieve specific performance goals.

Syntax

```
config seti <param>=<value>
```

Parameters

Parameter	Description
param	Name of the INI parameter
value	New value for the INI parameter

Example

```
$> config seti numbcast=3
OK
```

3.2.7 config reload

Description

Configuration information saved in "stored" memory is loaded into active memory. The system starts using the new loaded parameters immediately.

The keyword 'all' can be used in place of a *module* to reload all modules.

Syntax

```
config reload <module>|all
```

Parameters

Parameter	Description
module	reload the specified *module*

Example

```
$> config reload all
Done

Creating a new user and making it active would be done as follows...
$>  config set user mrdusty privilege=user password=mypw
Done

$> show user mrdusty
Error: Entry not found
$> config reload users
Done
$> show user mrdusty
User information:
   userId:      fred
   privilege:   USER
   description:
```

3.2.8 config restore

Description

Restore all configurations to their factory default settings.

This command will restore all settings to their factory default settings. All user settings will be lost and cannot be recovered.

The VManager must be explicitely restarted after issuing this command.

Syntax

```
config restore
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$>  config restore

OK
```

3.3 exec

The exec commands interact with the network or topology database.

3.3.1 exec clearStats

Description

Clear accumulated statistics

Syntax

```
exec clearStats
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> exec clearStats

Done
```

3.3.2 exec decommission

Description

This command prepares a mote for graceful removal from the network by moving its children to other parents. Returns a callback ID - this callback ID will be contained in a subsequent console notification when the mote is ready for deletion. Note that in some cases a child may have no other possible parents (this will never occur if following recommended deployment guidelines) so the child will be stranded when the mote is reset.

Syntax

```
exec decommission <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Decommission the mote specified by *macAddr* or *moteId*

Example

```
$>  exec decommission 00-17-0D-00-00-12-34-56

Done. Callback id: 1
```

3.3.3 exec deleteMote

Description

Delete a mote from the network. If the mote is not currently lost, it should be decommissioned first to avoid potential data loss.

This command should be used to clear a mote's join counter in the event that it is completely reflashed. Failure to do so will prevent a mote from joining the network until the join counter matches. The same applies to *blink* packets when using that mode, none will be delivered until the join counter matches.

Syntax

```
exec deleteMote <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Delete the mote specified by *macAddr* or *moteId*

Example

```
$>  exec deleteMote 00-17-0D-00-00-12-34-56

Done
```

3.3.4 exec deleteUnjoined

Description

Removes configdb entries for motes not currently in the topology. This command is useful for removing an unjoined mote that has been reprogrammed such that it's join counter may not match that in the configdb.

Syntax

```
exec deleteUnjoined
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$>  exec deleteUnjoined

Done
```

3.3.5 exec exchJoinKey

Description

This command changes the join key on the specified mote. This command also updates the corresponding ACL entry once the mote has responded affirmatively to the command.

Syntax

```
exec exchJoinKey <macAddr|moteId> <key>
```

Parameters

Parameter	Description
macAddr or moteId	Change the *joinKey* on the mote specified by *macAddr* or *moteId*
key	The new 16-byte join key

Example

```
$>  exec exchJoinKey 00-17-0D-00-00-12-34-56 000102030405060708 0A0B0C0D0E0F

Done. Callback id: 1
```

3.3.6 exec exchNetId

Description

This command distributes a new network ID to all the motes in the network. Returns a callback ID - this callback ID will be contained in a subsequent console notification when the netId has been changed.

Syntax

```
exec exchNetId <newNetId>
```

Parameters

Parameter	Description
newNetId	Change the network's ID to *newNetId*

Example

```
$>  exec exchNetId 294

Done. Callback id: 2
```

3.3.7 exec motelog

Description

This command retrieves a mote's reset log.

Syntax

```
exec motelog <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Retrieve the log file from the mote specified by *macAddr* or *moteId*

Example

```
$> exec motelog 00-17-0D-00-00-12-34-56
$> TRACE MOTELOG
   2015-12-09 15:10:27.890 from mote 00-17-0D-00-00-12-34-56 : (x100)
```

In this example the mote returns reset code 0x100 indicating a normal (watchdog) reset.

3.3.8 exec setAdv

Description

This command controls advertising in the network. Setting to *off* turns off all advertisements to prevent motes from joining the network. Setting to *on* turns on mote advertisements to allow motes to join the network. The VManager does not automatically turn advertising on and off.

> ⚠ It is dangerous to turn off advertising in the network. When advertising is off, new motes can not join and existing motes can not rejoin the network after a reset. Turning off advertising may be useful in unusual situations, such as to prevent motes from joining the network or to save power. In most cases, it is best to allow advertising to remain under the control of the VManager.

Syntax

```
exec setAdv <on|off>
```

Parameters

Parameter	Description
on or off	on = enable advertisements, off = disable advertisements

Example

```
$> exec setAdv on
Done
```

3.3.9 exec sendData

Description

Send a packet to a mote. This command is equivalent to invoking the API command POST /motes/m/{mac}/dataPacket.

Syntax

```
exec sendData <macAddr|moteId> <srcPort> <dstPort> <priority> <payload>
```

Parameters

Parameter	Description
macAddr or moteId	Send packet to the mote specified by *macAddr* or *moteId*
srcPort	UDP source port of the packet
dstPort	UDP destination port of the packet
priority	Priority of the packet. One of LOW, MEDIUM, or HIGH
payload	Payload bytes of the packet, in hexidecimal. Maximum size of payload is 80 bytes

Example

Send packet to mote, source port=0xF0B8, destination port= 0xF0B8 , priority=MEDIUM, payload bytes=0x11 0x 22 0x 33 0x44 0x55:

```
$> exec sendData 00-17-0D-00-00-12-34-56 61624 61624 MEDIUM 1122334455

Done. Callback id: 2
```

3.3.10 exec sendIP

Description

This command sends a packet with specified payload to a mote. It requires that the user construct a valid 6LoWPAN header that is prepended to their data. This command is equivalent to invoking the API command POST /motes/m/{mac}/ipPacket.

Syntax

```
exec sendIP <macAddr> <priority> <encryptionOffset> <payload>
```

Parameters

Parameter	Description
macAddr	MAC address of the mote that is the destination of the packet
priority	Priority of the packet. One of LOW, MEDIUM, or HIGH
encryptionOffset	The offset in bytes to the the start of the encrypted portion of the packet (normally the start of user data)
payload	Payload bytes of the packet, in hexidecimal. This consists of the prepended 6LoWPAN header followed by the user data. Maximum size of payload is 80 bytes

Example

To send packet from the manager to mote 00-17-0D-00-00-12-34-56 , with source port=0xF0B8, destination port= 0xF0B8 , priority=MEDIUM, and payload bytes=0x11 0x 22 0x 33 0x44 0x55:

- The encryption offset is 0.
- The IP header fields are as follows:

Field	Bytes	Contents
LOWPAN_IPHC	2	011.11.1.10:C.S.ss.M.D.dd • 011 - IPHC dispatch • 11 - Traffic Class and Flow Label are elided • 1 - Next Header field is compressed • 10 - Hop Limit is compressed • C (context identifier) = 0 (elided) • S (source compressed) = 1 • ss (source mode = 11 (elided) • M (multicast) = 0 (none) • D (destination compressed) = 1 • dd (destination mode) = 11 (elided) 0111.1110.0111.0111 = 0x7E77
Context ID Extension	0	Not used
Source Address	0	Elided
Destination Address	0	Elided
UDP Header	0/1	111101.S.D • S/D (source/dest. port is compressed) = 11 1111.0111 = 0xF7
UDP Ports	1	s.d • s (source port = F0Bs) = b1000 • d (dest port = F0Bd) = b1000 1000.1000 = 0x88

So the 6LoWPAN header is 0x7D77F788.

```
$> exec sendIP 00-17-0D-00-00-12-34-56 MEDIUM 0 7E77F77D77F7881122334455

Done. Callback id: 20
```

3.4 exit/logout/quit

Description

This command exits the console application, returning the user to the Linux shell. Can be invoked either as "exit", "logout" or "quit".

Syntax

```
exit | logout | quit
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$>  exit
```

3.5 help

Description

Show list of commands available, or details of a command.

Syntax

```
help [command] [subCommand]
```

Parameters

Parameter	Description
command	Any of the CLI commands

Example

```
$>  help
For more information on a specific command, type HELP command-name
clear           Clear screen
...

$> help clear
Usage: clear

Clear screen.
```

3.6 ping

Description

This command requests that the mote or AP indicated send a response containing reply time, temperature and voltage. This is an application layer command and does not use ICMP echo.

Syntax

```
ping <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Ping the mote specified by *macAddr* or *moteId*

Example

```
$> ping 1
Done. Callback id: 2
$> PING 2015-10-12 14:12:40.598 Reply from Mote #1, mac: 00-17-0d-00-00-12-34-56
  CallbackId: 2, Latency: 8ms (0 hops), Data: 3276mV, 34C
```

3.7 reset

The reset commands reset specific devices or the network

3.7.1 reset mote

Description

Issue reset to a mote or AP.

Syntax

```
reset <mote|ap> <macAddr|moteId>
```

Parameters

Parameter (one of the following)	Description
mote or ap	Reset device type specified by *macAddr* or *moteID*
macAddr or moteId	The device to be reset

Example

```
$>  reset mote 25

OK

$>  reset ap 00-01-02-03-04-05-06-07

OK
```

3.7.2 reset network

Description

Reset the network. An optional ‘reload’ argument indicates whether configuration should be reloaded prior to starting the network again.

Syntax

```
reset network [--reload]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$>  reset network --reload
Done
```

3.8 show

The show commands display current (active) configuration, statistics, and volatile information. To see persistent parameters, use *config* .

Commands that generate multi-page output can be invoked with a `-p | --page` switch to produce paginated output - pressing the **return** key advances the page.

3.8.1 show acl

Description

Displays a list of the motes currently whitelisted on the manager Access Control List. When the -p or --page option is used, output is paused after each page. When a MAC address is provided, only the entry for that device is printed.

Syntax

```
show acl [-p|--page] [macAddr]
```

Parameters

Parameter	Description
macAddr	Print only the device with MAC address *macAddr* if on the ACL

Example

```
$>  > show acl
ACL:

    MAC: 00-17-0D-00-00-38-FF-FF

    MAC: 00-17-0D-00-00-37-6E-A1
```

3.8.2 show alarms

Description

Returns a list of all open alarms.

Syntax

```
show alarms
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show alarms

Alarm info list:
  2015-03-17 16:34:38 Maximum number of motes reached
```

3.8.3 show time

Description

This command displays information on the AP Bridge Connector associated with the specified AP Bridge.

Syntax

```
show apc <macAddr|moteId>
```

Parameters

Parameter	Description
<macAddr|moteId>	Return information for the mote specified by *macAddr* or *moteId*

Example

```
$> show apc 1
APC apc-603528, Interface ID: 1
  AP #1, MAC: 00-17-0D-00-00-60-35-28 (Oper)
  Version:    1.0.1.16 (built 2016/08/04 11:43:34)
  State:      Working
  Connection: 127.0.0.1:41444

APC statistics:
  RX pkts:     119998
  TX pkts:     37149
  TX delays:   <5ms: 37149, <7ms: 0, <10ms: 0, <50ms: 0, >50ms: 0; Max delay: 4.743ms
  Pauses:      0
  Disconnects: 0
```

3.8.4 show apcs

Description

Displays the list of all Access Point Controllers (APCs) in the network. Each APC represents the connection of an AP Bridge to the VManager. When the -p or --page option is used, output is paused after each page.

Syntax

```
show apcs [-p|--page]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show apcs
Name        ID   APC St  AP MAC                   AP ID  AP St
----------- ---- ------- ----------------------- ------ -------
apc-603528     1 Working 00-17-0D-00-00-60-35-28       1 Oper
apc-603772     3 Working 00-17-0D-00-00-60-37-72       3 Oper
```

This command lists all the apcs currently or previously in the network.

- Name: APC identifier
- ID: index of AP
- APC St: Current state of the APC connection (**Working, Busy, Offline**)
- AP MAC: EUI-64 of the AP Mote
- AP ID: Short address assigned to the AP Mote by the manager
- AP St: Current state of the AP Mote (**Nego**, **Conn**, **Oper**, **Lost**)

3.8.5 show aps

Description

Displays the list of all Access Points (APs) in the network. This is similar to the show motes command to display the list of motes. When the -p or --page option is used, output is paused after each page.

Syntax

```
show aps [-p|--page]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show aps
AP MAC                  Id  Clk State  State time  Jn  Nbrs Links
----------------------- --- --- ----- ------------ --- ---- -----
00-17-0D-00-00-60-35-28   1 Int  Oper   0-20:06:36   1    4    60
00-17-0D-00-00-60-37-72   3 Net  Oper   0-07:26:08   2    4    28

APs: 2. Live: 2, joining: 0, lost: 0
```

This command lists all the aps currently or previously in the network.

- AP MAC: EUI-64 of the device
- ID: short address assigned to this device by the manager
- Clk: Clock source for the AP (**Int, Ext, Net**)
- State: Current state of each device (**Negot**, **Conn**, **Oper**, **Lost**)
- State time: Time (d-hh:mm:ss) since the device was advanced to its current state. When a device is **Operational**, State time shows how long it has been in the network
- Age: Seconds since the most recent packet was received by the manager from this device
- Jn: Shows how many times the device has joined and advanced to the **Operational** state
- Nbrs: Number of neighbors with which this device has active links
- Links: Total number of active links on this device

3.8.6 show dcl

Description

Dsiplays a list of the motes currently blacklisted (not allowed to join) on the manager Deny Control List. When the -p or --page option is used, output is paused after each page. When a MAC address is provided, only the entry for that device is printed.

Syntax

```
show dcl [-p|--page] [macAddr]
```

Parameters

Parameter	Description
macAddr	Print only the device with MAC address *macAddr* if on the DCL

Example

```
$>  > show dcl
DCL:

    MAC: 00-17-0D-00-0B-AD-0B-AD
```

3.8.7 show ini

Description

Displays currently used and "Active" ini settings. If a parameter *param* isn't specified or a n asterisk (*) is used, displays all parameters.

> ⚠ This command requires superuser privileges - see the su command.

Syntax

```
show ini [param | *]
```

Parameters

Parameter	Description
param	The ini *param* to be displayed. The "*" parameter will display all ini parameters

Example

```
#> show ini NUMBCAST
MNGRINI Module configuration :
NUMBCAST:2
```

3.8.8 show mote/ap

Description

This command returns network and neighbor information about the specified mote or Access Point (AP). When the -p or --page option is used, output is paused after each page. When the -a or --all-neighbors option is used, all neighbors (linked and discovered) are shown - normally only linked neighbors are shown. When the -l or --links option is used, information on each link is shown - normally on the total number of links per path are shown.

Syntax

```
show mote [-p|--page] [-a|--all-neighbors] [-l|--links] <macAddr|moteId>
show ap [-p|--page] [-a|--all-neighbors] [-l|--links] <macAddr|APId>
```

Parameters

Parameter	Description
macAddr or moteId	Return information for the mote specified by *macAddr* or *moteId*

Example

```
$> show mote 00-17-0D-00-00-DD-EE-FF
MOTE #2, MAC: 00-17-0D-00-00-DD-EE-FF
  Version: 1.3.2.4 (stack 1.2.3.6)
  State: Oper, Hops: 1.0, State time:   0-00:02:55, Age: 30
  Power: 65534 (Routing)
  Power Cost: Max 65534, FullTx 65, FullRx 65, Used 401
  Capacity: 200 links, 31 neighbors
  Number of neighbors (parents, children)   : 1 (1, 0)
  Bandwidth total / descendants (requested) : 14849 / 0 (27840)
  Number of links total, TX / RX / requested: 12, 3 / 0 / 1

Statistics:
  Reliability:     100.000% (0 lost, 15 total)
  Avg Latency:     1052 ms, 3712 ms est. to mote
  Voltage:         3306 mV
  Charge consumed: 29335 mC

Neighbors:
  #   1 parent Q:75 links: 3 rssi:-41/-41 Ready

$> show ap 1
AP #1, MAC: 00-17-0D-00-00-60-39-7F
  Version: 1.4.0.76 (stack 1.4.0.5)
  Identity: apc-60397f
  State: Oper, Hops: 0.0, State time:   1-21:13:27
  Clock Source: Int
  Capacity: 1000 links, 499 neighbors
  Number of neighbors (parents, children) : 7 (0, 7)
  Descendant bandwidth                    : 758
  Number of links total, TX / RX          : 50, NA / 30
Neighbors:
  #   2 child  Q: 94 links:   3 rssi:-47/-53
  #   3 child  Q: 95 links:   3 rssi:-58/-59
  #   4 child  Q: 96 links:   2 rssi:-42/-53
  #   5 child  Q: 92 links:  10 rssi:-54/-55
  #   6 child  Q: 96 links:   3 rssi:-45/-56
  #   7 child  Q: 91 links:   5 rssi:-50/-62
  #   8 child  Q: 93 links:   4 rssi:  0/-53
```

Description of the fields in the reply:

General:

- Mote #: moteId of the mote
- MAC: EUI-64 of the mote
- State: Current state - one of Idle, Negot1-2, Conn1-5, Oper, Lost
- Hops: Average ("empirical") hops for upstream data
- State time: Time since last state change
- Age: Time, in seconds, since the manager last received a packet from this device
- Power: Power, i.e. maxStCurrent in powerSrcInfo param reported by mote - one of Routing, or Low Power (maxStCurrent less than needed for routing). Note that "routing type" can be set either on the mote or on the manager. If either the mote or the manager declares a mote to be non-routing, then the mote will not be assigned children or advertisement links
- Power Cost: powerSrcInfo param reported by mote
- Capacity: Maximum number of links and neighbors that are supported by mote
- Number of neighbors: first entry is # parents + # children = # nbrs, first entry in parentheses is # parents, second entry in parentheses is # children
- Bandwidth (ms/packet): total (mote + descendants) / descendants, and requested for this mote alone in parentheses. A lower value here is more bandwidth. In this example the mote has more bandwidth than it requested
- Number of links: Total links across all slotframes, number of upstream TX links, number of upstream RX links, link equivalent of bandwidth requested by this mote

Statistics:

- Reliability: Percentage, lost and total generated in parentheses
- Avg Latency: Average upstream latency, estimated average downstream latency
- Voltage: Supply voltage in mV
- Charge consumed: Charge since last mote reset in mC

Neighbors:

- Neighbor moteId, relationship (parent or child or '-' for discovered), path quality in percent (30% or 75% until path stability is measured), number of upstream links to neighbor, RSSI to, RSSI from, and whether the path is in use (Ready), or merely discovered (Not Ready).

3.8.9 show motes

Description

Displays the list of all motes in the network. When the -p or -- page option is used, output is paused after each page.

Syntax

```
show motes [-p|--page]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show motes
Mote MAC                Id  State  State time  Age Jn  Nbrs Links
----------------------- --- ----- ------------ --- --- ---- -----
00-17-0D-00-00-30-4B-1D   2  Oper   0-20:10:32  12   1    4    20
00-17-0D-00-00-30-44-A4   4  Oper   0-20:09:56  14   1    4    20
00-17-0D-00-00-30-5A-BD   5  Oper   0-20:09:53   0   1    4    45

Motes: 3. Live: 3, joining: 0, lost: 0

$> show aps
AP MAC                  Id  Clk State  State time  Jn  Nbrs Links
----------------------- --- --- ----- ------------ --- ---- -----
00-17-0D-00-00-60-35-28   1 Int  Oper   0-20:11:28   1    4    60
00-17-0D-00-00-60-37-72   3 Net  Oper   0-07:31:00   2    4    28

APs: 2. Live: 2, joining: 0, lost: 0
```

This command lists all the motes currently or previously in the network.

- MAC: EUI-64 of the device
- MoteID: short address assigned to this device by the manager
- State: Current state of each device (**Negot**, **Conn**, **Oper**, **Lost**)
- State time: Time (d-hh:mm:ss) since the device was advanced to its current state. When a device is **Operational**, State time shows how long the mote has been in the network
- Age: Seconds since the most recent packet was received by the manager from this mote
- Jn: Shows how many times the device has joined and advanced to the **Operational** state

- Nbrs: Number of neighbors with which this device has active links
- Links: Total number of active links on this device

3.8.10 show network

Description

Displays information about the network configuration and statistics.

Syntax

```
show network
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show network
Network configuration:
   networkId:                1229
   topologyType:             MESH
   downFrameMultiplierDelay: 3600000
   ccaMode:                  False
   ipAddrPrefix:             FE80::
   basePkPeriod:             15000
   downFrameMultiplier:      1
   joinSecurityType:         COMMON_SKEY
   minServicePkPeriod:       100
   downFrameSize:            512
   numParents:               2
   channelList:              32767
   upFrameSize:              1024
Network statistics:
   Network start time:  2015-12-04 15:13:42.408, uptime   4-01:32:49
   Live motes:          12
   Reliability:         100.000% (0 lost, 477682 total)
   Avg Latency:         1168 ms
   Path stability:      81.164%
   Advertising:         On
   Queue (net/user):    0/0
   Current frame size:  512
```

Network configuration

See `config set network parameters` for details

Network statistics

- Network start time: start time and uptime
- Live motes: Number of motes (not APs) in the **Operational** state
- Reliability: The network reliability is calculated as the percentage of packets generated by any mote that are successfully received by the manager. This number should be 99.99% or higher in a healthy network. This statistic is counted directly on the manager by monitoring the security counter on the packets as they come in
- Avg Latency: The average upstream latency of packets received by the manager. The manager checks the packet header for the generation timestamp (ASN) and compares this to the current ASN to calculate each individual packet latency
- Path stability: The network path stability represents the percentage of total MAC transmissions that have succeeded. This number will vary depending on individual mote placement. The network is designed to achieve 100% reliability even at 50% stability. At lower stability values, motes will use more energy and bandwidth as more packets need to be retried. This statistic is computed based on health reports received from motes. The manager increments the total number of transmits and fails with each new health report arrival
- Advertising: Current state of advertising
- Queue: Number of packets in the manager queue, both for manager generated (net) and user generated packets
- Current frame size: Downstream frame size (slots) at this time, the downstream frame can lengthen in steady-state if downFrameMultiplier > 1

3.8.11 show paths

Description

This command displays information all paths for a specified mote. When the -p or --page option is used, output is paused after each page. The -a or --all option additionally lists unused potential paths.

Syntax

```
show paths [-p|--page] [-a|--all] <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	The *macAddr* or *moteId* of device

Example

```
$> show paths 00-17-0D-00-00-AA-BB-CC
Neighbors:
  #   2 child  Q:82 links: 3 rssi:-41/-41 Ready
  #   3 child  Q:75 links: 4 rssi:  0/  0 Not Ready
```

This command displays all the paths between the specified device and its active neighbors. Each row of the display lists the mote ID of a neighbor, its relationship (Parent or Child), the number of links assigned to that path, the path quality statistics, and the RSSI.

In this example, mote 2 is a child of this mote and the has a Quality which has been measured based on success/fail statistics to be 82% (path stability), has 3 links assigned, and has a RSSI of -41dB.

3.8.12 show route

Description

Displays the source route from the manager to the mote specified.

Syntax

```
show route <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Display the source route to the mote specified by *macAddr* or *moteId.*

Example

```
S> show route 20
Route for Mote #20, 00-17-0D-00-00-38-00-D9
   -> AP   #   1, 00-17-0D-00-00-60-36-D7
   -> MOTE #   7, 00-17-0D-00-00-38-00-55
   -> MOTE #  11, 00-17-0D-00-00-38-00-B6
   -> MOTE #  14, 00-17-0D-00-00-38-00-C0
   -> MOTE #  17, 00-17-0D-00-00-38-00-C9
```

Each hop gets one row of output. In this example, the route to moteId=20 is 1-7-11-14-17-20.

3.8.13 show services

Description

Displays service information for the specified mote.

Syntax

```
show services <macAddr|moteId>
```

Parameters

Parameter	Description
macAddr or moteId	Display service information for the mote specified by *macAddr* or *moteId*

Example

```
$> show services 00-17-0D-00-00-DD-EE-FF
Services:
  #   0 MAC: 00-17-0D-00-00-00-FF-FE
        Allocated BW: 27840
        Latency:      1737 (1 hops)
```

3.8.14 show sessions

Description

Displays the list of current user sessions. A session is created when a user logs into Console or authenticates in the External API.

> The External API will generate a new user session whenever an API request contains the Authentication header. This can cause a large number of sessions to be listed in the *show sessions* output.
>
> Internally, the number of user sessions is limited to a maximum number based on activity, older sessions will be removed if the number of sessions grows too large.

Syntax

```
show sessions
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show sessions
User           Last activity            Login                    Client Info
-------------  -----------------------  -----------------------  ---------------
dust           2016-05-24 11:10:44.542  2016-05-24 11:10:44.542
```

3.8.15 show system

Description

Display system information such as the the system start and uptime, the user designated system name string, the user designated location string, and the CLI timeout. These fields can be set using the *config set system* command.

Syntax

```
show system
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show system
System information:
   System start: 2015-12-04 15:13:42.433, Uptime:   3-23:15:12
   sysName:     Thermal1
   location:    Unit5
   cliTimeout:  0 minutes
```

3.8.16 show time

Description

Displays two different measures of time: Server system time, and network (AP) time.

Syntax

```
show time
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show time
System start : 2015-12-01 15:10:22.627, System uptime :   0-19:43:07
Network start: 2015-12-01 16:50:40.717, Network uptime:   0-18:02:49
```

3.8.17 show unjoined

Description

Display motes that have a join counter stored in the configdb, but are not part of current topology.

Syntax

```
show unjoined
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show unjoined
TODO
```

3.8.18 show user

Description

Display information about a specified user.

Syntax

```
show user <userId>
```

Parameters

Parameter	Description
userID	Display information for the user specified by *userId.*

Example

```
$> show user dust
User information:
   Id:        dust
   Password:  ****
   Privilege: USER
   Description: duster
```

3.8.19 show users

Description

Display information about all users. When the -p or --page option is used, output is paused after each page.

Syntax

```
show users [-p|--page]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> show users
User configuration:
Id:     dust
```

3.8.20 show ver

Description

Displays the versions of the individual software components of the VManager, as well as the VManager package.

Syntax

```
show ver [component]
```

Parameters

Parameter	Description
component	Leave blank for *all* or specify the component name. In most cases, the component name matches the name used in the output. Use *apiserver* to refer to the External API component.

Example

```
$> show ver
Package version: 1.0.1.6dev
External API version: 1.0.1.6
AuthManager version: 1.0.1.6 (built 2016/03/01 16:26:49)
ConfigDB version: 1.0.1.6 (built 2016/03/01 16:28:09)
Console version: 1.0.1.6
Manager version: 1.0.1.6 (built 2016/03/01 16:13:21)
Watchdog version: 1.0.1.6 (built 2016/03/01 16:28:58)


$> show ver console
Console version: 1.0.1.6
```

3.9 sm

Description

This command shows a list of all motes and Access Points (APs) in the network. It is similar to the show motes or show aps commands.

Syntax

```
sm [-p|--page]
```

Parameters

Parameter	Description
	This command has no parameters

Example

```
$> sm

AP MAC                  Id  Clk State  State time  Age Jn  Nbrs Links
----------------------- --- --- ----- ------------ --- --- ---- -----
00-17-0D-00-00-60-37-9C   1 Int  Oper   0-00:01:29 112   1    3    15

Mote MAC                Id  State  State time  Age Jn  Nbrs Links
----------------------- --- ----- ------------ --- --- ---- -----
00-17-0D-00-00-30-08-2D   2  Oper   0-00:03:02  11   1    1    10
00-17-0D-00-00-30-08-05   3  Nego   0-00:00:02  23   1    0    0

APs: 1, Motes: 2, 1 live, 1 joining
```

This command lists all the devices currently or previously in the network.

- MAC: EUI-64 of the device
- MoteID: short address assigned to this device by the manager
- Clk: Clock source (APs only)
- State: Current state of each device (**Nego**, **Conn**, **Oper**, **Lost**)
- State time: Time (d-hh:mm:ss) since the device was advanced to its current state. When a device is **Operational**, State time shows how long the mote has been in the network
- Age: Seconds since the most recent packet was received by the manager from this mote. Age may grow large for APs.
- Jn: Shows how many times the device has joined and advanced to the **Operational** state
- Nbrs: Number of neighbors with which this device has active links
- Links: Total number of active links on this device

3.10 su

Description

Enter superuser mode, which enables some commands primarily used for debugging and testing. The console prompt changes from $> to #> to indicate that the user is in superuser mode.

Syntax

```
su becareful
```

Parameters

Parameter	Description
	This command takes no parameters

Example

```
$>  su becareful
#>
```

3.11 trace

Description

Enable/disable trace output to console. Before this setting will take effect, the set loglevel command must be used to activate the trace. The full set of traces is available by typing the subscribe command, but most of these are intended only for internal debugging purposes. The following traces are of general interest:

TraceId	Description
mngr.net.io.data	User data packets
mngr.tplgdb.mote	Mote state changes (e.g. motes progressing through the join process)
mngr.tplgstat	Upstream latency and hops information

Syntax

```
set loglevel <traceId> TRACE
trace [<traceId> <on|off>]
```

Parameters

Parameter	Description
traceId	Enable the *traceId* trace
on or off	on = enable trace, off = disable trace

Example

```
$> set loglevel mngr.tplgstat TRACE
Done
$> trace mngr.tplgstat on
Trace enabled for mngr.tplgstat

2015-12-15 15:59:06.698 mngr.tplgstat: L_TRACE Stat. Mote #12 ASN packet: 58559826934 current:
58559827083
2015-12-15 15:59:06.702 mngr.tplgstat: L_TRACE Stat. Mote #12 AP#1 Hops. New hops: 50 new average:
64
2015-12-15 15:59:06.705 mngr.tplgstat: L_TRACE Stat. Mote #12 Latency. New latency: 1080250 usec.
New average: 871756 usec.
```

In this example, the manager has received a packet from moteId 12. The manager compares the current ASN to the ASN contained in the packet header to calculate the upstream latency in µs. The manager uses the TTL field in the header to calculate the number of hops taken to get to the AP, and this value is printed in units of 0.1 hops. Above, the packet hops are specified as 50 indicating that the packet took 5 hops to reach the AP. This trace also prints an IIR-filtered average of the upstream latency and hops for this mote.

Trademarks

Eterna, Mote-on-Chip, and SmartMesh IP, are trademarks of Dust Networks, Inc. The Dust Networks logo, Dust, Dust Networks, and SmartMesh are registered trademarks of Dust Networks, Inc. LT, LTC, LTM and  are registered trademarks of Linear Technology Corp. All third-party brand and product names are the trademarks of their respective owners and are used solely for informational purposes.

Copyright

Disclaimer

This documentation is provided “as is” without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

This documentation might include technical inaccuracies or other errors. Corrections and improvements might be incorporated in new versions of the documentation.

Linear Technology does not assume any liability arising out of the application or use of any products or services and specifically disclaims any and all liability, including without limitation consequential or incidental damages.

Linear Technology products are not designed for use in life support appliances, devices, or other systems where malfunction can reasonably be expected to result in significant personal injury to the user, or as a critical component in any life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Linear Technology customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify and hold Linear Technology and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Linear Technology was negligent regarding the design or manufacture of its products.

Linear Technology reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products or services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to Dust Network's terms and conditions of sale supplied at the time of order acknowledgment or sale.