

# SmartMesh WirelessHART Manager API Guide

# Table of Contents

---

1	About This Guide	6
1.1	Related Documents	6
1.2	Conventions Used	8
1.3	Revision History	9
2	Introduction	10
2.1	Control Channel	10
2.2	Notification Channel	11
3	Protocol	12
3.1	XML	12
3.2	XML-RPC	12
3.3	Control Channel Message Format	13
3.3.1	Logging in to the Control Channel	13
3.3.2	Control Session Limits	15
3.4	Data Representation	15
3.4.1	Data Types	15
3.5	Notification Channel Message Format	17
3.5.1	Subscribing to the Notification Channel	17
3.5.2	Notification Session Limits	19
4	Commands	20
4.1	activateAdvertising	20
4.2	activateFastPipe	21
4.3	cancelOtap	23
4.4	cli	24
4.5	deactivateFastPipe	26
4.6	decommissionDevice	27
4.7	deleteConfig	29
4.8	exchangeJoinKey	30
4.9	exchangeMoteJoinKey	32
4.10	exchangeMoteNetworkId	34
4.11	exchangeNetworkId	36
4.12	exchangeNetworkKey	37
4.13	exchangeSessionKey	39
4.14	getConfig	41
4.15	getLatency	43
4.16	getLicense	44
4.17	getTime	45
4.18	login	46
4.19	logout	47
4.20	pingMote	48

4.21	promoteToOperational	49
4.22	reset	50
4.23	resetWithId	51
4.24	resetWithMac	52
4.25	sendRequest	53
4.26	sendResponse	55
4.27	serviceRequest	57
4.28	setConfig	59
4.29	setLicense	61
4.30	setNumParents	62
4.31	startOtap	63
4.32	startOtapWithRetries	64
4.33	subscribe	65
4.34	unsubscribe	67
5	Configuration Document	68
5.1	getConfig	68
5.2	setConfig	73
5.3	deleteConfig	74
5.4	Acl	75
5.5	Network	76
5.6	Network Statistics	81
5.7	Motes	83
5.8	Mote Statistics	86
5.9	OtapStatus	88
5.10	OtapStatus Mote	89
5.11	Redundancy	90
5.12	Security	91
5.13	Sla	92
5.14	System	93
5.15	Paths	95
5.16	Path Statistics	97
5.17	Users	99
5.18	SourceRoute	100
5.19	Open Alarms	101
6	Notifications	102
6.1	Data	102
6.2	Events	104
6.2.1	netPathCreate	104
6.2.2	netPathDelete	105
6.2.3	netPathActivate	106
6.2.4	netPathDeactivate	107
6.2.5	netPathAlert	108
6.2.6	netPipeOn	109

6.2.7	netPipeOff	110
6.2.8	netMoteJoin	111
6.2.9	netMoteJoinQuarantine	112
6.2.10	netMoteLive	113
6.2.11	netMoteUnknown	114
6.2.12	netMoteDisconnect	115
6.2.13	netMoteJoinFailure	116
6.2.14	netMoteInvalidMIC	117
6.2.15	netPacketSent	118
6.2.16	netServiceDenied	119
6.2.17	netPingReply	121
6.2.18	sysBootUp	122
6.2.19	sysConnect	123
6.2.20	sysDisconnect	124
6.2.21	sysManualMoteReset	125
6.2.22	sysManualMoteDelete	126
6.2.23	sysManualMoteDecommission	127
6.2.24	sysManualNetReset	128
6.2.25	sysManualDccReset	129
6.2.26	sysManualStatReset	130
6.2.27	sysConfigChange	131
6.2.28	sysCmdFinish	133
6.2.29	sysRdntModeChange	135
6.2.30	sysRdntPeerStatusChange	136
6.2.31	netReset	137
6.2.32	netTransportTimeout	138
6.2.33	netMoteQuarantine	139
6.3	Alarms	140
6.3.1	alarmOpen	141
6.3.2	alarmClose	142
6.4	Log and Error Messages	143
6.5	CLI Notifications	144
6.6	Health Reports	145
7	Definitions	148
7.1	Advertising Status	148
7.2	AP Redundancy Mode	148
7.3	Application Domain	148
7.4	Bandwidth Profile	149
7.5	Channel Type	149
7.6	Response Codes	149
7.7	Frame State	151
7.8	Mote OTAP Status	152
7.9	Mote State	152

---

7.10 Object Type	153
7.11 OTAP State	153
7.12 Packet Priority	154
7.13 Path Direction	154
7.14 Pipe Direction	154
7.15 Pipe Status	154
7.16 Reason	155
7.17 Redundancy Peer Status	155
7.18 Redundancy Mode	156
7.19 Redundancy Mode Reason	156
7.20 Reset Object	156
7.21 Security Mode	157
7.22 Timeout Type	157
7.23 User Privilege	157

# 1 About This Guide

---

## 1.1 Related Documents

---

The following documents are available for the SmartMesh WirelessHART network:

Getting Started with a [Starter Kit](#)

- [SmartMesh WirelessHART Easy Start Guide](#) - walks you through basic installation and a few tests to make sure your network is working
- [SmartMesh WirelessHART Tools Guide](#) - the Installation section contains instructions for the installing the serial drivers and example programs used in the Easy Start Guide and other tutorials.

User Guide

- [SmartMesh WirelessHART 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.

Interfaces for Interaction with a Device

- [SmartMesh WirelessHART Manager CLI Guide](#) - used for human interaction with a Manager (e.g. during development of a client, or for troubleshooting). This document covers connecting to the CLI and its command set.
- [SmartMesh WirelessHART Manager API Guide](#) - used for programmatic interaction with a manager. This document covers connecting to the API and its command set.
- [SmartMesh WirelessHART 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 WirelessHART Mote API Guide](#) - used for programmatic interaction with a mote. This document covers connecting to the API and its command set.

Software Development Tools

- [SmartMesh WirelessHART Tools Guide](#) - describes the various evaluation and development support tools included in the [SmartMesh SDK](#) including tools for exercising mote and manager APIs and visualizing the network.

Application Notes

- [SmartMesh WirelessHART Application Notes](#) - app notes covering a wide range of topics specific to SmartMesh WirelessHART networks and topics that apply to SmartMesh networks in general.

Documents Useful When Starting a New Design

- The Datasheet for the [LTC5800-WHM SoC](#), or one of the [castellated modules](#) based on it, or the backwards compatible [LTP5900 22-pin module](#).
- The Datasheet for the [LTP5903-WHR](#) embedded manager.
- A [Hardware Integration Guide](#) for the mote SoC or [castellated module](#), or the [22-pin 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 program firmware on a device.
- ESP software - used to program firmware images onto a mote or module.
- Fuse Table software - used to construct the fuse table as discussed in the Board Specific Integration Guide.

#### Other Useful Documents

- A glossary of wireless networking terms used in SmartMesh documentation can be found in the [SmartMesh WirelessHART User's Guide](#).
- 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`



## 1.3 Revision History

---

Revision	Date	Description
1	07/16/2012	Initial Release
2	03/18/2013	Numerous Small Changes
3	10/22/2013	Commands and schema elements added for redundancy, advertising, parents
4	04/04/2014	Added description of the contents of the Dust Device and RSSI Health Reports;
5	10/28/2014	Clarified that reset is required for network configuration elements to update; Updated deleteConfig command; Clarified decommissionDevice command; Clarified packet format in sendRequest command; Added to exchangeMoteJoinKey description; Added description of Dust RSSI Report; Other minor changes
6	04/22/2015	Clarified subscription filter usage
7	12/03/2015	Minor changes

## 2 Introduction

---

This guide describes the commands used to communicate with the LTP5903CEN-WHR SmartMesh WirelessHART manager through the XML API. The API is intended for machine-to-machine communications (e.g. a host program talking to the manager).

In contrast, the command line interface (CLI) is intended for human interaction with a manager, e.g. during development, or for interactive troubleshooting. See the [SmartMesh WirelessHART Manager CLI Guide](#) for details on that interface.

The SmartMesh WirelessHART Manager functions as the server that provides an interface between a client application and a WirelessHART network. It presents an Application Programming Interface (API) that is encapsulated in XML and is contacted via an Ethernet network. In the XML API, the Manager uses two channels for communication with a client: a two-way control channel that permits requests and responses between the manager and a client, and a one-way notification channel that streams data from the manager to a client.

- The Control channel provides a client with a Remote Procedure Call (RPC) interface for querying Manager status, updating configuration information, and interacting with the mote network. The Control channel uses the XML-RPC protocol.
- The Notification channel is a persistent TCP/IP stream over which the Manager sends notifications to the client in an XML format.

### 2.1 Control Channel

---

The manager and client applications use the control channel to exchange commands and information about a WirelessHART network. The first command from the client to the manager is a login, and the manager returns a token that the client uses in all subsequent commands. All communications over the control channel use the format described by the XML-RPC API, and take the form of HTTP requests.

By default, the control channel port is set to 4445.


The control channel is used to make API calls to query the Manager and network state and make configuration changes. For example, the control channel is used to:

- Send a data packet to a mote
- Retrieve the list of motes in the network
- Retrieve the list of neighbors of a mote
- Retrieve network statistics
- Configure the network ACL
- Activate a high-bandwidth connection (pipe) to a mote.

## 2.2 Notification Channel

---

The manager uses the notification channel to stream data and network events to the client. To subscribe to the notification channel, clients use a subscribe method call over the control channel, and the manager returns a token and the port number at which the notification channel is available. Clients then initiate a TCP connection to the manager on the notification port, and send an authorization request containing the token to the manager over the TCP connection. As long as the connection remains open, the manager returns a stream of XML sensor data and network events.

 Since the notification channel is used for streaming data, it uses XML messages over a TCP/IP connection. The notification channel does not use XML-RPC.

Client scripts or applications listen on the notification channel and process the XML-wrapped data. The notification channel contains:

- Data sent upstream from a mote
- Network events such as mote joins and resets

Communications over the notification channel use the format described by the Notification XML schema. See “[Notification Channel Message Formats](#)” for a general overview, and see the “[Notifications](#)” section for a reference guide.

## 3 Protocol

---

The WirelessHART Manager API is built on top of simple, standard message formats. Client applications use XML and XML-RPC to communicate with the manager. The Manager API is composed of a Control channel for RPC-style configuration and command functions and a Notification channel for asynchronous notifications of network events.

### 3.1 XML

---

XML is a self-describing, human-readable, extensible metalanguage that uses descriptive tags to define the organizational structures and items in a data set or document. The SmartMesh manager XML API defines tags that describe the components and configuration of SmartMesh-enabled networks. For example, the following XML stanza encapsulates network data about a live SmartMesh-enabled network:

```
<config>
  <Network>
    <netName>myNet</netName>
    <networkId>1</networkId>
    <optimizationEnable>true</optimizationEnable>
    <numMotes>100</numMotes>
    <maxMotes>200</maxMotes>
  </Network>
</config>
```

### 3.2 XML-RPC

---

XML-RPC is a simple protocol that enables a computer to execute remote procedures on another computer using HTTP and XML. In a typical XML-RPC exchange, a client computer uses HTTP to send an XML document containing a method name and arguments to a server. The server invokes the method with the arguments, and then wraps up the return value of the method in another XML document, which it sends back to the client.

The SmartMesh manager implements a number of methods for querying and setting configuration of the manager and the network. Most of these methods interact with a configuration document on the manager. This configuration document is defined by the SmartMesh manager XML API schemas.



The XML-RPC server on the manager uses chunked HTTP transfers for large amounts of data.

For more information about XML-RPC, see the following Web sites:

- <http://www.xmlrpc.com/> (specification, information, and examples)
- <http://ws.apache.org/xmlrpc/index.html> (Apache implementation)
- <http://www.dom4j.org/index.html> (Java open-source XML framework)
- <http://xmlrpc-c.sourceforge.net/example-code.php>. (C examples)
- <http://www.jmarshall.com/easy/http/> (chunked HTTP transfers)

## 3.3 Control Channel Message Format

The format for control channel messages uses XML-RPC. XML-RPC defines request and response formats in XML for making function calls over the HTTP protocol. The HTTP messages are composed of a headers followed by a message body. The body of the HTTP request contains an XML document containing the method name and parameters. The body of the HTTP response contains an XML document containing the method response or a fault message. XML-RPC parameters are passed as XML elements in a list. XML-RPC defines a small set of native parameter types, as well as struct and array containers for building more complex types.

XML-RPC libraries for many languages are available that interoperate with the Manager.



### Interoperability notes

The Manager implementation of XML-RPC does not handle HTTP Keep-Alive connections. The Manager XML API requires that each XML-RPC method call occurs in its own HTTP connection. If an XML API client tries to perform an XML-RPC call in an existing HTTP connection, the Manager may disconnect the connection without responding to the subsequent call.

### 3.3.1 Logging in to the Control Channel

To begin interacting with the control channel, a client application logs into the manager. If you simply want to log in and begin receiving data, use the instructions in this section to log in to the control channel and then skip ahead to "Subscribing to the Notification Channel".

The login XML-RPC call is shown in detail below. If the client is using an XML-RPC library, the client:

1. Establishes a connection to the manager IP address and control channel port (by default, port 4445).
2. Calls the login method with the username and password.
3. Stores the login token from the login response for use in further communication.

## Login Request

```
POST /RPC2 HTTP/1.0
Host: 10.10.16.126:4445
User-Agent: xmlrpclib.py/1.0.1 (by www.pythonware.com)
Content-Type: text/xml
Content-Length: 219

<?xml version='1.0'?>
<methodCall>
  <methodName>login</methodName>
  <params>
    <param><value><string>admin</string></value></param>
    <param><value><string>admin</string></value></param>
  </params>
</methodCall>
```

## Login Response

```
HTTP/1.1 200 OK
Connection: Close
Content-Type: text/xml
Content-Length: 167
X-Powered-By: ulxmlrpcpp/1.7.4
Server:
Date: Wed Apr 25 15:15:28 2012

<?xml version="1.0" encoding="utf-8"?>
<methodResponse>
  <params>
    <param>
      <value><string>login-token</string></value>
    </param>
  </params>
</methodResponse>
```

When the client is finished with API activity, the client calls [logout](#) to release the login token.

### 3.3.2 Control Session Limits

The manager allows up to 100 concurrent control sessions. Control sessions are initiated with the Manager API *login* function. Control sessions must be released with the Manager API *logout* function. If an additional user tries to login, the manager tries to find sessions that haven't been used in at least 10 minutes. If any inactive sessions are found, the session that has been inactive the longest is closed to make room for the new one. If no inactive sessions are found, the new user's login call will return an error. The oldest session is measured based on the time of the last command sent using the session's login token. There is no way to query the list of tokens from the manager. So if a client disappears without logging out, there's no way to recover the client's token and perform a proper logout. An administrator can release tokens using the CLI command.

## 3.4 Data Representation

Each of the API Commands shows a list of parameters to be passed into the XML-RPC call in the *Request* section and the parameters that will be returned by the call in the *Response* section. For details on how parameters are passed in an XML-RPC call, see the [XML-RPC specification](#).

The Configuration Document is an XML string that is sent as a query or response from certain API commands, e.g. *getConfig*, *setConfig* and *deleteConfig*. When passed as a parameter of the XML-RPC *methodCall* or *methodResponse*, the XML characters of this string must be escaped to avoid interference with the XML-RPC protocol.

The API Notifications are not transmitted over XML-RPC, API Notifications are XML data sent over a TCP socket (the Notification channel). Each of the API Notifications shows a list of the elements that are present in the XML notification string.

### 3.4.1 Data Types

The XML-RPC specification defines the encodings for a small number of primitive types.

Type	Description
integer	4-byte signed integer value in decimal format, corresponds to the XML-RPC <code>int</code> type.
float	Floating point value, corresponds to the XML-RPC <code>double</code> type.
string	String value. Special XML characters (<, > and &) passed in a string value must be escaped.
boolean	Boolean value represented by the strings <code>1</code> or <code>true</code> or <code>0</code> or <code>false</code> .

The Manager API uses some more specific encodings for certain types of values. These parameters are passed as a native XML-RPC type, but the Manager requires that the value is in a more specific format.

Type	XML-RPC Type	Description
macAddress	string	EUI-64 identifier, or MAC address, represented as a hyphen-separated list of 8 two-character hex values, e.g. 00-17-0D-00-00-01-02-03.
hex data	string	Sequence of two-character hex values, e.g. 0102A3A4 represents the bytes 0x01 0x02 0xA3 0xA4.
timestamp	integer	Time represented as the number of milliseconds since 00:00:00.000 1/1/1970 GMT.

In addition, enumerated types are represented as string parameters, which accept a limited number of values.

### Parameter example

For example, a command named `example` with the following Request description, would be expected to pass the `methodCall` shown below in the XML-RPC call.

#### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of a mote
arrivalTime	timestamp		Time of arrival
payload	hex data		Payload

```

<methodCall>
  <methodName>example</methodName>
  <params>
    <param><value><string>my-token</string></value></param>
    <param><value><string>00-01-02-03-04-05-06-07</string></value></param>
    <param><value><int>1429742699</int></value></param>
    <param><value><string>8081828384858687</string></value></param>
  </params>
</methodCall>

```



## 3.5 Notification Channel Message Format

---

### Notification Channel Message Format

The notification channel is a long-lived TCP connection in contrast to the short-lived HTTP connections used by the control channel. The notification channel formats messages as XML, see the [Notifications section](#).

A notification session is initiated by a client making a *subscribe* call on the control channel. The manager returns a response with the notification port and a notification token, client creates a TCP connection to the notification port on the manager, sends the authorization message, and the manager pushes notifications asynchronously to the client over the TCP connection.

### 3.5.1 Subscribing to the Notification Channel

#### Subscribe Request

```
POST /RPC2 HTTP/1.0
Host: 10.10.16.126:4445
User-Agent: xmlrpc.py/1.0.1 (by www.pythonware.com)
Content-Type: text/xml
Content-Length: 228

<?xml version="1.0" ?>
<methodCall>
  <methodName>subscribe</methodName>
  <params>
    <param><value><string>login-token</string></value></param>
    <param><value><string>all</string></value></param>
  </params>
</methodCall>
```

## Subscribe Response

```
HTTP/1.1 200 OK
Connection: Close
Content-Type: text/xml
Content-Length: 251
X-Powered-By: ulxmlrpcpp/1.7.4
Server:
Date: Wed Apr 25 15:50:08 2012

<?xml version="1.0" encoding="utf-8"?>
<methodResponse>
  <params>
    <param><value><array><data>
      <value><string>notif-token</string></value>
      <value><i4>24112</i4></value>
    </data></array></value></param>
  </params>
</methodResponse>
```

The client creates a TCP connection to the manager on port 24112 and sends:

```
<dustnet><authrq><token>notif-token</token></authrq></dustnet>
```

If the notification token is accepted, the manager replies with the start of the notification document:

```
<dustnet version="1.0">
<authrs/>
```

The `dustnet` element is not terminated. All of the data and event messages are contained in the `dustnet` element. This allows a client to parse the notification stream as elements within a single document object.

The manager sends further notifications to the client:

```
<event>
  <timeStamp>1335394208207</timeStamp>
  <eventId>15242</eventId>
  <sysConnect>
    <userName>admin</userName>
    <ipAddr>10.10.48.22</ipAddr>
    <channel>notif</channel>
  </sysConnect>
</event>

<data>
  <moteId>9</moteId>
  <macAddr>00-1B-1E-00-00-00-08</macAddr>
  <time>1335394453601</time>
  <payload type='80'>0000030d03000009fc0201db</payload>
</data>
```

When the client closes the connection, the manager closes the notification document:

```
</dustnet>
```

## 3.5.2 Notification Session Limits

The manager allows up to 5 concurrent notification sessions. Notification sessions are created by a *subscribe* request on the control channel, but notification channel requires the client to connect and authenticate before any notification data is sent.

The notification session is closed under any of the following conditions:

- The client sends an *unsubscribe* request on the control channel with the associated notification token.
- The client closes the TCP connection of the notification session.
- If the client does not read notifications from the TCP connection quickly enough, the manager's write queue to the TCP connection will fill up and the manager will close the TCP connection.

## 4 Commands

### 4.1 activateAdvertising

#### Description

This command triggers the manager to activate advertising for a specified mote or all motes. The advertising status of each mote is reported in the Mote configuration element, which is accessible with the [getConfig](#) command.

#### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of mote on which advertising is to be activated. The special value <code>FF-FF-FF-FF-FF-FF-FF-FF</code> activates advertising on all motes.
timeout	integer		The number of minutes (from 0 to 255) that advertising should remain on. After this period, advertising is turned off. Specifying 0 turns off advertising immediately.

#### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the manager accepted the command and will attempt to turn on advertising; "OK" does not signify that advertising was turned on. An error is returned if the command cannot be executed.

#### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.
API_OBJECT_NOT_FOUND	The object was not found

## 4.2 activateFastPipe

### Description

This command triggers the manager to activate a high-bandwidth connection (pipe) between the manager and a mote. Only one pipe may be active in the network at a time (see Network Bandwidth Control). A `netPipeOn` notification is sent when the pipe is successfully turned on. The pipe status of a mote can be checked using the mote configuration's `pipeStatus` element.

The actual pipe bandwidth allocated by the manager depends on power information reported by the destination mote and the motes through which the pipe is built. To determine the allocated pipe bandwidth, use the `getConfig` command to retrieve a mote's configuration and check the `allocatedPipePkPeriod` element.

A pipe can be upstream only, downstream only, or bi-directional.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of pipe destination
pipeDirection	string	<a href="#">Pipe Direction</a>	Pipe direction

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the manager accepted the command to activate the pipe. "OK" does not signify that the pipe was activated.  An error is returned if the command cannot be executed.

### Response Codes


Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_VAL_CANT_CREATE	Validation error. Cannot create the object because it already exists.

API_VAL_ENUM	Validation error. The value was not in the enumeration.
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.

## 4.3 cancelOtap

### Description

This command cancels the OTAP (Over-The-Air-Programming) process to upgrade software on motes and the access point.

 This command is not valid if the OTAP process is in the "committing" stage in which motes are switching over to the new software. You can query the OTAP status using the [getConfig](#) command.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the OTAP process was cancelled. If OTAP was not cancelled, a response code indicating an error is returned.


### Response Codes


Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full

## 4.4 cli

### Description

This command tunnels a given command through to the manager's Command Line Interface (CLI). The *cli* command can be called by only one XML API client at a time. The output to the given *cli* command is tunneled back to the client via the notifications channel. To receive the command output, the client must be subscribed to *cli* notifications (see [Notification Channel](#)).

 The *cli* command may return a failure message ("Could not write to CLI session") if the client is not subscribed to *cli* notifications.

 The *cli* command is deprecated. Use direct API functions instead of tunneling through the Manager CLI (or invoke the CLI from a command line session).

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
command	string		CLI command to be tunneled

### Response

Parameter	Type	Enum	Description
result	string		'OK' indicates that the command is successfully tunneled.  An an error is returned if the command was not successfully tunneled.

### Response Codes

Code	Description
API_CLI_NULL_COMMAND	Empty CLI command



---

API_CLI_WRITE_ERROR	Could not write to the CLI session
XML_INVALID_AUTHENTICATION	The command could not be authenticated

## 4.5 deactivateFastPipe

### Description

This command triggers the manager to start deactivating a high-bandwidth channel between the manager and specified mote. The `netPipeOff` notification is sent when the pipe has been successfully turned off.



This command can only deactivate a pipe that was activated by the manager API. Do not call this command to deactivate a pipe that was activated as a result of a service request from a network device.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of pipe destination.

### Response

Parameter	Type	Enum	Description
result	string		<p>"OK" indicates the manager accepted the command to deactivate the pipe.</p> <p>"OK" does not signify that the pipe was activated.</p> <p>An error is returned if the command cannot be executed.</p>

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found

## 4.6 decommissionDevice

### Description

The *decommissionDevice* command causes the manager to prepare a mote (identified by its MAC address) for safe removal from the network. The target mote's children are moved to other parents, and the node is designated as non-routing (it does not forward any other mote's traffic). This makes it safe to delete the target mote without disrupting other network traffic. The mote enters the *disconnecting* state on the Manager, which can be used to inform a user that it is safe to power down the mote and physically remove it from the network. Note that the *decommissionDevice* command can be used on any mote unlike the *deleteConfig* command, which can only be used to remove a lost or disconnected mote.

A *sysManualMoteDecommission* event is generated when the manager starts the process of decommissioning a mote. A *netMoteDisconnect* event is generated when the mote has been decommissioned and can be safely removed from the network.



Even after the *netMoteDisconnect* event is received, the mote remains on the network as a leaf node until the mote is powered down or reset.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		The MAC address of the device to be decommissioned

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates that the command was received and the manager will attempt to decommission the mote.

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated

API_OBJECT_NOT_FOUND	The object was not found
----------------------	--------------------------

## 4.7 deleteConfig

### Description

The *deleteConfig* command updates the manager configuration document to remove the element specified by the *configQuery* parameter. Only a limited set of elements can be deleted:

- Mote - In versions prior to 4.1, mote elements can only be deleted if the mote is not in the network
- User
- ACL

The *configQuery* must have the structure matching a section of the [Configuration Document](#).

### Request

Parameter	Type	Enum	Description
loginToken	string		Login token for this session
configQuery	string		Document expression describing which elements in the configuration document to delete. The <i>configQuery</i> is an XML string that must be escaped to be properly encapsulated in the XML-RPC request.

### Response

Parameter	Type	Enum	Description
OK	string		'OK' is returned if the command is successful

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The login token is not valid for an active control session
API_OBJECT_NOT_FOUND	The object specified in the <i>configQuery</i> could not be found
XML_PARSE_ERROR	The <i>configQuery</i> contains an invalid XML string

## 4.8 exchangeJoinKey

### Description

This command triggers the manager to initiate a change of common join key for all motes in the network. The join key is a symmetric encryption key that is used by the manager and motes to encrypt and decrypt the join messages that they exchange when the mote is attempting to join the network. The *exchangeJoinKey* command cannot be used if the manager is in *acceptACL* mode or if it is in *acceptCommonJoinKey* mode and there are entries in the Access Control List (ACL).

The join key is exchanged without loss of data, and no resetting of motes is needed. The execution of the exchange is set in the future to allow time for the command to propagate across the network and allow time for re-transmissions. The manager generates a *sysCmdFinish* event when the command is synchronously executed by the motes. The delay is typically tens of minutes after the *exchangeJoinKey* command is issued and depends on the network size and configuration. For networks using the P1 configuration, the delay is ~21 minutes. During this period no additional *exchangeJoinKey* commands may be issued. Motes that reset during the exchange may or may not receive the new join key. If a mote does not receive the new join key it cannot re-join the network.



Note that network motes must be in the Operational state when the *exchangeJoinKey* command is issued in order for them to receive the new join key. For best results, use the following procedure:

1. Before exchanging the join key, first determine if all network motes are operational by issuing the *getConfig config/Motes/Mote/state* command. If there are non-operational motes, wait for them to rejoin the network. If they do not rejoin within an hour, troubleshoot the problem (for example, check the mote batteries).
2. When all motes are operational, issue the *exchangeJoinKey* command.
3. When the *sysCmdFinish* event is received, reissue the *getConfig config/Motes/Mote/state* command to determine if all motes are operational. Do one of the following depending on the outcome:
  - If all motes are operational, they have all received the new join key and no further action is needed.
  - If some motes are non-operational, wait up to an hour to see if the missing motes can rejoin the network. If the missing motes do not rejoin, repeat steps 2 and 3 with the old network join key so that these motes can rejoin. Then repeat steps 1-3 with the new join key.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
newKey	hex data		New common join key as 32-character hexadecimal string (representing 16 bytes)

## Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, the manager returns the callback ID for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback ID.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>

## Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.
API_VAL_ENCKEY	Validation error. The encryption key must contain exactly 32 hex characters.
API_OBJECT_NOT_FOUND	The object was not found

## 4.9 exchangeMoteJoinKey

### Description

This command triggers the manager to initiate the update of the join key for a specific mote (identified by its MAC address) and update the manager's Access Control List (ACL). See the Network Security section of the [SmartMesh WirelessHART User's Guide](#) for more information. The target mote needs to be reset after the successful exchange for the new join key to take effect.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		The MAC address of the mote to which the manager is distributing the join key
newKey	hex data		New key as 32-character hexadecimal string (representing 16 bytes)

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, manager returns the callback id for this request. When the mote has replied to the manager, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback id and an error code indicating whether the mote successfully completed the command.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.
API_VAL_ENCKEY	Validation error. The encryption key must contain exactly 32 hex characters.




API_OBJECT_NOT_FOUND	The object was not found
----------------------	--------------------------

## 4.10 exchangeMoteNetworkId

### Description

This command initiates an update of the Network ID for a given mote. See the Network Security section of the [SmartMesh WirelessHART User's Guide](#) for more information.

 HART recommends that some network IDs be reserved for specific purposes- see HCF Spec 75, table 2. 0xFFFF is never a valid network ID.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		The MAC address of the mote to which the manager is distributing the new Network ID
newId	integer		New Network ID for this mote

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, manager returns the callback id for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback id.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>

### Response Codes


Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.

API_VAL_MINMAX	Validation error. The maximum value must be greater than the minimum value.
API_OBJECT_NOT_FOUND	The object was not found.

## 4.11 exchangeNetworkId

### Description

This command initiates an update of the Network ID for all motes in the network. See the [Network Security](#) section of the [SmartMesh WirelessHART User's Guide](#) for more information.

 HART recommends that some network IDs be reserved for specific purposes- see HCF Spec 75, table 2. 0xFFFF is never a valid network ID.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
newId	integer		New Network ID

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, the manager returns the callback id for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback id.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>


### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_MINMAX	Validation error. The maximum value must be greater than the minimum value.
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.

## 4.12 exchangeNetworkKey

### Description

This command triggers the manager to generate a new network key (link layer authentication key) and distribute it to the motes. The network key is exchanged without loss of data, and no resetting of motes is needed. The execution of the exchange is set in the future to allow time for the command to propagate across the network and allow time for re-transmissions. The manager generates a *sysCmdFinish* event when the command is synchronously executed by the motes. The delay is typically tens of minutes after the *exchangeNetworkKey* command is issued and depends on the network size and configuration. For networks using the P1 configuration, the delay is ~21 minutes. During this period no additional *exchangeNetworkKey* commands may be issued. Note that if a mote resets during the exchange, it will receive the new network key when it re-joins the network.

 The network key is a network-wide symmetric authentication key that is shared by the manager and the motes in its network. The network key is used to authenticate all messages other than join requests, activation messages, and advertisements on the data link (MAC) layer. The manager gives the network key to motes when they join the network

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, the manager returns the callback ID for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback ID.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>


### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.

## 4.13 exchangeSessionKey

### Description

This command triggers the manager to generate and distribute a new random session key for the session between the manager (or gateway) and a mote. The session key is used to encrypt all messages exchanged between the manager (or gateway) and a mote. When the session key exchange is completed, a *sysCmdFinish* event is generated. The session key is exchanged without loss of data, and no resetting of motes is needed.

 The time required to complete the command and issue a *sysCmdFinish* notification may vary depending on the size and type of the network. During this period no additional *exchangeSessionKey* commands may be issued.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddrA	macAddress		MAC address of the Manager (F980000000000001) or Gateway (F981000000000002) to identify which session a new session key is to be generated
macAddrB	macAddress		MAC address of the mote for which the new session key is to be generated. Specifying the broadcast address (FF-FF-FF-FF-FF-FF-FF) will update the broadcast key for all motes.

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, the manager returns the callback ID for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> and <i>sysConfigChange</i> event notifications that include the corresponding callback ID.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned</p>

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full



## 4.14 getConfig

### Description

The *getConfig* command retrieves a section of the manager configuration. The configuration is represented as an XML document with several top-level children:

- System
- Network
- Motes
- Paths
- Users
- Security

Each element of the configuration document contains fields containing configuration parameters or child elements. The structure of the configuration document is described in more detail in the [Configuration Document](#) section.

### Request

Parameter	Type	Enum	Description
loginToken	string		Login token for this session
depth	string		Integer value (or <i>all</i> ) indicating the depth of the response (depth of child elements to expand)
configQuery	string		Document expression describing which elements in the configuration document to retrieve. The <i>configQuery</i> is an XML string that must be escaped to be properly encapsulated in the XML-RPC request.

For example, the *configQuery* string for a specific mote is constructed as:

```
<config> <Motes> <Mote> <macAddr>00-17-0d-00-00-01-02-03</macAddr> </Mote> </Motes> </config>
```

The corresponding XML-RPC call looks like:

```

<methodCall>
  <methodName>getConfig</methodName>
  <params>
    <param><value><string>my-login-token</string></value></param>
    <param><value><string>all</string></value></param>

    <param><value><string>&lt;config&gt;&lt;Motes&gt;&lt;Mote&gt;&lt;macAddr&gt;00-17-0d-00-00-01-02-03&lt
  </params>
</methodCall>

```

More examples can be found in the [Configuration Document](#) section.

## Response

Parameter	Type	Enum	Description
configDocument	string		Returns the requested configuration document. The <i>configDocument</i> is an XML string that must be escaped to be properly encapsulated in the XML-RPC response.

## Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The login token is not valid for an active control session
API_OBJECT_NOT_FOUND	The query did not any configuration elements
XML_PARSE_ERROR	The <i>configQuery</i> contains an invalid XML string

## 4.15 getLatency

### Description

This command returns an estimate of the latency to communicate with a specific mote (identified by its MAC address). The return value is an array of two integers representing downstream latency and upstream latency, measured in milliseconds. The estimate applies only to the packet's over-the-air time, and does not include time it may spend in manager's queues prior to being sent.

Note that the latency calculations take into account long-term bandwidth, such as bandwidth set by the manager API or requested by service requests, but not short-term bandwidth or pipe bandwidth.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of the mote

### Response

Parameter	Type	Enum	Description
downstreamLatency	integer		Data latency (in milliseconds) for transmissions from the manager to mote
upstreamLatency	integer		Data latency (in milliseconds) for transmissions from mote to the manager

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found

## 4.16 getLicense

---

### Description

This command retrieves the license key used for enabling additional manager features. See [Administering the Manager](#) for more information about Software Licensing.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session

### Response

Parameter	Type	Enum	Description
license	hex data		License key as a 13-byte (26 character) hex string

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated

## 4.17 getTime

---

### Description

This command requests the current time as Coordinated Universal Time (UTC) and ASN (Absolute Slot Number).

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session

### Response

Parameter	Type	Enum	Description
utc_time	float		The number of seconds since Jan 1, 1970 in UTC. The fractional component provides microsecond precision.
asn_time	integer		Time as Absolute Slot Number (ASN)

### Response Codes

Code	Description
API_OBJECT_NOT_FOUND	The object was not found

## 4.18 login

---

### Description

The *login* command validates the username and password against the Manager configuration. If the username and password are valid, the Manager creates a login session on the control channel and returns a login token for the session. Subsequent commands from the client must contain a valid login token. The session remains active until the client calls the *logout* command with the login token.

### Request

Parameter	Type	Enum	Description
username	string		User name
password	string		User password

### Response

Parameter	Type	Enum	Description
loginToken	string		The token to be used in subsequent commands

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The username and/or password did not match a user

## 4.19 logout

---

### Description

The *logout* command ends a control channel login session. After the *logout* command, the client can no longer use the login token to execute control channel commands.

### Request

Parameter	Type	Enum	Description
login token	string		The login token for the session

### Response

Parameter	Type	Enum	Description
OK	string		The string "OK" is returned if the command is successful

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The login token is not associated with an active login session.

## 4.20 pingMote

### Description

This command sends a packet to a mote requesting a response. When the manager receives a reply from the mote, it will issue a *netPingReply* event notification that includes this callback ID.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		The MAC address of the mote to ping

### Response

Parameter	Type	Enum	Description
callbackId	integer		If the packet is accepted, the manager returns the callback ID for this request. If the packet was not accepted by the manager, a response code indicating an error is returned

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full



## 4.21 promoteToOperational

### Description

(Added in Manager 4.1.0) The *promoteToOperational* command is used to transition a **Quarantined** mote to the **Operational** state. The mote must be in the Manager's ACL for the promotion to succeed.

See the Network Security section of the SmartMesh WirelessHART Manager User Guide for details on Quarantine Mode.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		The MAC address of the Quarantined mote.

### Response

Parameter	Type	Enum	Description
result	string		<p>If the mote was transitioned to operational, manager returns the string "OK". A netMoteLive event will be generated when the mote's state is Operational.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated.
API_OBJECT_NOT_FOUND	The mote is not in the network.
API_VAL_STATE	The mote is in an invalid state
API_NO_ACL_ENTRY	The mote is not on the ACL

## 4.22 reset

### Description

This command resets the system or network, or clears a configuration object. There are several forms of the reset command based on what is being reset.

- *system* – reset the manager software
- *network* – reset the network
- *stat* – reset the manager's statistics
- *eventLog* – clear the manager's event log

The other form of reset, [resetWithMac](#), is used for resetting a mote.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
object	string	<a href="#">Reset Object</a>	Identifies the object to be reset

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the object was reset for all objects except a mote. If the object was not reset, a response code indicating an error is returned.


### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
XML_PARSE_ERROR	A parsing error occurred
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full

## 4.23 resetWithId

### Description

This command resets a mote, identified by its Mote ID. This command is always sent as a best-effort packet. The manager sends the [sysManualMoteReset](#) notification when it sends the reset command to a mote.

 This command is deprecated. Use the *resetWithMac* command to reset a mote.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
object	string		Identifies the object to be reset; for this command, the value must be the literal string "mote"
moteld	string		Mote ID of the mote to be reset

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the reset command was sent to the mote.

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
XML_PARSE_ERROR	A parsing error occurred
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full

## 4.24 resetWithMac

### Description

This command resets a mote, identified by its MAC address. The *resetWithMac* command is always sent as a best-effort packet. The manager sends the *sysManualMoteReset* notification when it sends the reset command to a mote.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
object	string		Identifies the object to be reset; for this command, the value must be the literal string "mote"
macAddr	macAddress		MAC address of the mote to be reset

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the reset command was sent to the mote

### Response Codes


Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
XML_PARSE_ERROR	A parsing error occurred
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full

## 4.25 sendRequest

### Description

Send a request packet to a mote. The *sendRequest* call returns a unique callback ID, which is used in the *netPacketSent* notification to indicate when the packet has been sent to the network and in the *data* notification when a mote sends a response packet.

The packet may either be a reliable request packet to initiate request/response exchange with the mote (reliable is `true`) or an unreliable packet that does not require an acknowledgement by the mote (reliable is `false`). The binary data is sent as a string in which each byte is represented as a two-digit hexadecimal number. The maximum data payload is 78 bytes.

 For non HART-compliant applications, messages must be prepended with a 4-byte header, 00 00 FC 12, leaving 74 usable bytes.

If the manager's internal queue is full, it responds with the response code `API_NETLAYER_FULL`. In this case, the application should retry after a short timeout.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of mote to which the packet is to be sent. FFFFFFFFFFFFFFFF can be used to broadcast to all motes.
domain	string	Application Domain	Application domain specifies the type of packet to be sent to the mote
priority	string	Packet Priority	Specifies the packet priority
reliable	boolean		Set to <code>true</code> if this is part of an request/response transaction. Set to <code>false</code> if this is part of a best-effort (unacknowledged) transaction. Note that a broadcast packet must be sent as best-effort transaction.
data	hex data		Packet payload, sent as a hexadecimal string with a maximum length of 78 or 74 bytes.

**i** A HART command in the data field begins with a 1-byte status, a 1-byte extended status, a 2-byte command ID, a 1-byte command length, followed by the command payload. Refer to HCF Spec-85 and 155 for details.

E.g. to send a command 796 to read the keep-alive timer, the following bytes would be sent (assuming status and extended status are 0's): 00 00 03 1E 01 02

## Response

Parameter	Type	Enum	Description
callbackId	integer		<p>Callback ID. If the packet is accepted by the manager and queued for transmission, the manager returns a unique identifier, callback id, for this request. After the manager sends the packet into the network, it issues a <i>netPacketSent</i> event notification that includes the callback id.</p> <p>If the <i>sendRequest</i> is a reliable transaction, the manager also includes the callback id when it sends the client the reply data packet it receives from the mote.</p> <p>If the packet was not accepted, a response code indicating an error is returned.</p>

## Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_VAL_ENUM	Validation error. The value was not in the enumeration.
API_OBJECT_NOT_FOUND	The object was not found
API_NETLAYER_FULL	The manager's transmit queue is full
API_VAL_CROSSREF	Validation error. The cross reference failed.
API_VAL_PATTERN_HEX	Validation error. The value contained an invalid hexadecimal character (valid characters are a-f, A-F, and 0-9).
API_VAL_DATAPACKET	Validation error. The message does not fit in the packet.

## 4.26 sendResponse

### Description

This command sends a response packet to a mote (identified by its MAC address). The parameters are similar to the parameters of the [sendRequest](#) command. The payload data is sent as a string in which each byte is represented as a two-character hexadecimal number.

Only unreliable responses are supported.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
macAddr	macAddress		MAC address of mote to which the packet is to be sent. FFFFFFFFFFFFFFFF can be used to broadcast to all motes.
domain	string	<a href="#">Application Domain</a>	Application domain specifies the type of packet to be sent to the mote
priority	string	<a href="#">Packet Priority</a>	Specifies the packet priority
reliable	boolean		Must be set to <code>false</code> , only unreliable responses are supported
callbackId	integer		Reserved; must be set to 0
data	hex data		Packet payload, sent as a hexadecimal string with a maximum length of 74 bytes

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>Callback ID. If the packet is accepted by the manager and queued for transmission, the manager returns a unique identifier for this response. After the manager sends the packet into the network, it issues a <code>netPacketSent</code> event notification that includes the callback id.</p> <p>If the packet was not accepted, a response code indicating an error is returned.</p>

**Response Codes**

<b>Code</b>	<b>Description</b>
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_VAL_ENUM	Validation error. The value was not in the enumeration.
API_NETLAYER_FULL	The manager's transmit queue is full
API_VAL_CROSSREF	Validation error. The cross reference failed.
API_VAL_PATTERN_HEX	Validation error. The value contained an invalid hexadecimal character (valid characters are a-f, A-F, and 0-9).
API_VAL_DATAPACKET	Validation error. The message does not fit in the packet.



## 4.27 serviceRequest

### Description

(Added in Manager 4.1.0) Requests that the Manager create a downstream service to the destination mote.

The caller should choose a Service ID when creating a service. The Service ID must be unique for services on the destination mote, but the same Service ID value can be used for different destinations. The service can be modified by calling *serviceRequest* with the same Service ID and updated parameters. A service can be deleted by specifying a *period* of 0.

In the 4.1.0 Manager, at most one service can be created to each destination mote.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
srcMac	macAddress		The MAC address of the source mote for the service request. This parameter must be set to the MAC address of the Gateway (F981000000000002).
destMac	macAddress		The MAC address of the destination mote for the service request.
serviceId	integer		Unique identifier for this service to the destination mote.
domain	App Domain		The domain for the service request.
period	integer		interval between packets, in ms
isSource	boolean		SrcMac will be generating packets. Must be set to True.
isSink	boolean		Reserved, set to FALSE
isIntermittent	boolean		Reserved, set to FALSE

### Response

Parameter	Type	Enum	Description
callbackId	integer		<p>If the command is accepted, manager returns the callback id for this request. When the exchange is completed, the manager issues a <i>sysCmdFinish</i> event notification that includes the corresponding callback id.</p> <p>If the command was not accepted by the manager, a response code indicating an error is returned.</p>

**Response Codes**

<b>Code</b>	<b>Description</b>
XML_INVALID_AUTHENTICATION	The command could not be authenticated.
API_OBJECT_NOT_FOUND	The destination mote was not found.
API_INVALID_MOTE	The source mote was not set to the MAC address of the gateway.
API_SERVICE_DENIED	The service request was denied.

## 4.28 setConfig

### Description

The *setConfig* command updates the manager configuration document with the contents of the *configDocument* parameter. The *configDocument* must have the structure matching a section of the [Configuration Document](#).

### Request

Parameter	Type	Enum	Description
loginToken	string		Login token for this session
configDocument	string		Configuration document containing the identifier elements and elements to update in the manager configuration. The <i>configDocument</i> is an XML string that must be escaped to be properly encapsulated in the XML-RPC request.

### Response

Parameter	Type	Enum	Description
configDocument	string		Returns the updated configuration section. The <i>configDocument</i> is an XML string that must be escaped to be properly encapsulated in the XML-RPC response.

For example, to set the *maxMotes* field of the Network element, the *configDocument* should contain:

```
<config>
  <Network>
    <maxMotes>176</maxMotes>
  </Network>
</config>
```

To set fields in an individual Mote element, the *configDocument* should contain:

```

<config>
  <Motes> <Mote>
    <macAddr>00-17-0d-00-00-01-02-03</macAddr>
    <name>HVAC_7</name>
    <powerSource>battery</powerSource>
  </Mote> </Motes>
</config>

```

## Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The login token is not valid for an active control session
API_OBJECT_NOT_FOUND	The object specified in the <i>configDocument</i> could not be found
XML_PARSE_ERROR	The <i>configDocument</i> contains an invalid XML string

## 4.29 setLicense

---

### Description

This command sets the license key, which controls whether certain features are available. See [Administering the Manager](#) for more information about Software Licensing.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
license	hex data		License key as a 13-byte (26 character) hex string

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the license was accepted and will be used after the Manager software is restarted

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_LICENSE	The license is not valid for this device

## 4.30 setNumParents

---

### Description

Added in Manager 4.1.0. Update the number of parents currently used by the Manager. During optimization, the Manager will attempt to assign this number of parents to each mote.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
numParents	integer		New number of parents for each mote.

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the manager accepted the command. An error is returned if the command cannot be executed.


### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated

## 4.31 startOtap

### Description

This command initiates the OTAP (Over-The-Air-Programming) process to upgrade software on motes and the Access Point. By default, the process will retry the OTAP file transmission 100 times.

 Before using the `startOtap` command, you need to copy the software upgrade image(s) onto the manager's OTAP directory, `/opt/dust-manager/otap`. You can use standard SSH-based tools to copy the OTAP file(s) onto the manager.

Once the OTAP process is started, it can be monitored using the `getConfig` command to retrieve the `OtapStatus` configuration element.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates the OTAP process was started. If OTAP was not started, a response code indicating an error is returned.


### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_VAL_STATE	Validation error. The current state does not allow this action to be performed.

## 4.32 startOtapWithRetries

### Description

This command initiates the OTAP (Over-The-Air-Programming) process to upgrade software for motes and the Access Point, using the specified number of retries.

 Before using the *startOtap* command, you need to copy the software upgrade image(s) onto the manager's OTAP directory, `/opt/dust-manager/otap`. You can use standard SSH-based tools to copy the OTAP file(s) onto the manager.

Once OTAP is started, it can be monitored using the *getConfig* command to retrieve the *OtapStatus* configuration element.

### Request

Parameter	Type	Enum	Description
token	string		Login token for this session
retries	integer		Number of times the manager attempts to resend the OTAP file into the network.

### Response

Parameter	Type	Enum	Description
result	string		"OK" indicates automatic OTAP process was started. If OTAP was not started, a response code indicating an error is returned.

### Response Codes

Code	Description
XML_INVALID_AUTHENTICATION	The command could not be authenticated
API_OBJECT_NOT_FOUND	The object was not found
API_VAL_STATE	Validation error. The current state does not allow this action to be performed.



## 4.33 subscribe

### Description

The *subscribe* command is issued over the control channel and creates (or updates) a notification session for the client. The *filter* parameter is a space-separated string with any combination of the following:

Notification Type	Description
all	Includes all notifications except log messages
data	Includes all data notifications
events	Includes all event notifications (event-alarm + event-network + event-system)
event-alarm	Includes alarm open and close events (all items listed under <a href="#">Alarms</a> )
event-network	Includes network events, such as link add, mote join ( <a href="#">Events</a> starting with "net")
event-system	Includes system events, such as config change, resets ( <a href="#">Events</a> starting with "sys")
sysError	Includes system error messages (see <a href="#">Log and Error Messages</a> )
log	Includes log messages (see <a href="#">Log and Error Messages</a> )
std-report	<i>Added in version 4.1.0</i> Includes standard HART health reports
vendor-report	<i>Added in version 4.1.0</i> Includes Dust health reports
hr	<i>Added in version 4.1.0</i> Includes all health reports
cli	Includes CLI notifications. For backwards compatibility only - not intended for new designs, however subscribing to 'all' will include CLI notifications.

If the token is a control channel login token, a new notification channel is created. If the token is a notification channel token (returned from a previous subscribe call), the subscription filter for the specified notification channel is updated.

### Request

Parameter	Type	Enum	Description
token	string		Login token or previously issued notification token
filter	string		Notifications to which the client wants to subscribe. The client can specify more than one notification type (from the table above) in this string by separating the values with spaces.

## Response

Parameter	Type	Enum	Description
token	string		Token to use for notification session
port	string		Port to use for notification session. The port value is only valid for new notification channels.

## Response Codes

Code	Description
API_VAL_ENUM	Validation error. The value was not in the enumeration.
XML_INVALID_NOTIF_CLIENT	The notification client does not exist.
XML_TOO_MANY_NOTIF_CLIENTS	The maximum number of notification clients has been reached.

## 4.34 unsubscribe

---

### Description

The *unsubscribe* command shuts down an existing notification channel. This is an alternative to the client closing the TCP connection. When the *unsubscribe* command is used to close a notification channel, the manager will send the closing element of the `dustnet` document element. The manager will then close the TCP connection.

### Request

Parameter	Type	Enum	Description
token	string		Notification token for the subscribe session.

### Response


Parameter	Type	Enum	Description
result	string		"OK" indicates the notification session will be shut down. If the connection was not closed, a response code indicating an error is returned.

### Response Codes

Code	Description
XML_INVALID_NOTIF_CLIENT	The notification client does not exist.

## 5 Configuration Document

The manager configuration and network topology is accessed and updated through the *getConfig*, *setConfig* and *deleteConfig* methods. These methods operate on elements in an XML document structure that is accessed by providing a document fragment that identifies a particular element in the document's schema.

 Each of the config methods (*getConfig*, *setConfig*, and *deleteConfig*) take a parameter that is an XML document query. Since the query string is encapsulated in an XML-RPC call, the XML delimiter characters of the query string must be escaped. Generally, this encapsulation is handled automatically by the XML-RPC library. However, if you are constructing the XML-RPC parameters manually, make sure to escape the query string as shown in the *getConfig* example below. Similarly, the XML document returned in a response will be escaped.

### 5.1 getConfig

The *getConfig* method takes two parameters to specify the response.

The depth parameter specifies the level to which child elements are expanded in the response. Setting the depth parameter to `all` shows all children. However, statistics must be queried separately from their container element. Specifying a depth of `1` requests values for the element's immediate children. This depth is generally used either to collect lists of elements or to collect all the values of a single element without recursing into all child elements.

The *configQuery* parameter is a query string that identifies an element by providing the XML path to the element. The following elements can be queried:

- System
- Network
  - Network configuration
  - Network statistics
  - Network service level (SLA)
- Motes
  - List of all motes
  - Individual motes, identified by MAC address
  - Mote statistics for individual motes
- Paths
  - List of all paths to neighbors for a particular mote
  - Path statistics for individual paths
- Security
  - Network security configuration
  - Access Control List (ACL) of allowed motes

- Users
- Alarms
- OTAP status
- Source Routes
- Redundancy status

For example, to retrieve the System element, the client calls the *getConfig* method with the following parameters:

Parameter name	Value
loginToken	my-login-token
depth	all
configQuery	<config><System/></config>

Within the TCP connection, the call would look like:

```

POST /RPC2 HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: 314
Host: 10.10.16.126:4445

<?xml version="1.0" ?>
<methodCall>
<methodName>getConfig</methodName>
<params>
  <param><value><string>dn40202fa3-66</string></value></param>
  <param><value><string>all</string></value></param>
  <param><value><string>&lt;config&gt;&lt;System&gt;&lt;/System&gt;&lt;/config&gt;</string></value></par

```

And the response looks like:

```

HTTP/1.1 200 OK
Connection: Close
Content-Type: text/xml
Content-Length: 556
X-Powered-By: ulxmlrpcpp/1.7.4
Server:
Date: Wed Apr 25 16:58:48 2012

<?xml version="1.0" encoding="utf-8"?>
<methodResponse>
  <params>
    <param><value><string>&lt;config&gt;&lt;System>
&lt;systemName>Test1&lt;/systemName>
&lt;location>Hayward&lt;/location>
&lt;swRev>4.0.0.21-6&lt;/swRev>
&lt;hwModel/>
&lt;hwRev/>
&lt;serialNumber>00170D80000D&lt;/serialNumber>
&lt;time>1335398328652&lt;/time>
&lt;startTime>1333583899000&lt;/startTime>
&lt;cliTimeout>1000&lt;/cliTimeout>
&lt;controllerSwRev>4.0.0.21&lt;/controllerSwRev>
&lt;/System&lt;/config&lt;/string&lt;/value&lt;/param>
  </params>
</methodResponse>

```



For simplicity, in the remainder of the examples, the XML query and response are shown without the HTTP header and XML escapes.

A query string identifies an element by providing the XML path to the element. For example, the Network element is queried as follows:

Parameter name	Value
loginToken	my-login-token
depth	all
configQuery	<config><Network/></config>

Because the depth parameter is `all`, the response contains the full content of the Network element, including the *ChannelBlacklist* and *Sla* children.

```

<config><Network>
  <netName>myNet</netName>
  <networkId>182</networkId>
  <optimizationEnable>true</optimizationEnable>
  <maxMotes>251</maxMotes>
  <numMotes>0</numMotes>
  <accessPointPA>true</accessPointPA>
  <ccaEnabled>false</ccaEnabled>
  <requestedBasePkPeriod>100000</requestedBasePkPeriod>
  <minServicesPkPeriod>400</minServicesPkPeriod>
  <minPipePkPeriod>480</minPipePkPeriod>
  <bandwidthProfile>P1</bandwidthProfile>
  <manualUSFrameSize>1024</manualUSFrameSize>
  <manualDSFrameSize>256</manualDSFrameSize>
  <manualAdvFrameSize>128</manualAdvFrameSize>
  <netQueueSize>0</netQueueSize>
  <userQueueSize>0</userQueueSize>
  <locationMode>off</locationMode>
  <ChannelBlackList>
    <frequency>2480</frequency>
  </ChannelBlackList>
  <Sla>
    <minNetReliability>99.00</minNetReliability>
    <maxNetLatency>12500</maxNetLatency>
    <minNetPathStability>50.00</minNetPathStability>
    <apRdntCoverageThreshold>70.00</apRdntCoverageThreshold>
  </Sla>
</Network></config>

```

An element within a container can be specified by filling in one of the identifier fields. For example, a mote can be queried as follows:

Parameter name	Value
loginToken	my-login-token
depth	all
configQuery	<config><Motes><Mote><macAddr>00-17-0D-00-00-38-01-02</macAddr></Mote></M

The depth parameter specifies the level to which child elements are expanded in the response. Setting the depth parameter to `all` shows all children. However, statistics must be queried separately from their container element. For example, lifetime Network statistics can be queried as follows:

Parameter name	Value
----------------	-------

loginToken	my-login-token
depth	all
configQuery	<config><Network><Statistics><lifetime/></Statistics></Network></config>



## 5.2 setConfig

The *setConfig* method takes a *documentString* parameter to specify the fields to be updated. The *documentString* should only contain the fields that are needed to identify the element and the fields that the client is updating. The configuration document description identifies the fields that are user-writable. The manager performs validation of the updated data before applying it to the configuration. In some cases, the manager must be restarted for the new configuration to take effect.

For example, the *maxMotes* parameter of the Network element is updated as follows:

Parameter name	Value
loginToken	my-login-token
configDocument	<config><Network><maxMotes>100</maxMotes></Network></config>

For cases where the client is updating an element with many instances (such as a Mote), the client must specify an identifier for the element as well as the parameter(s) to be updated. For example, the *name* parameter of a Mote element is updated as follows:

Parameter name	Value
loginToken	my-login-token
configDocument	<config><Motes><Mote><macAddr>00-1B-1E-00-00-00-00-08</macAddr><name>V.

## 5.3 deleteConfig

The *deleteConfig* method takes a query string parameter to specify the element to be removed.

The following configuration elements can be removed:

- Mote
- User
- ACL

An inactive Mote that's no longer part of the network can be removed from the manager's list of motes as follows:

Parameter name	Value
loginToken	my-login-token
configQuery	<config><Motes><Mote><macAddr>00-1B-1E-00-00-00-08</macAddr></Mote></M

An ACL entry can be removed as follows:

Parameter name	Value
loginToken	my-login-token
configQuery	<config><Security><Acl><Device><macAddr>00-1B-1E-00-00-00-08</macAddr>

For additional exploration of the Manager API, use the API Explorer tool from the SmartMesh SDK.

## 5.4 Acl

---

### Description

The *Acl* element is a child element of *Security*. The *Acl* element contains a list of Device elements with the following fields, one for each mote in the network ACL. See Network Security for more information about managing join keys.

The *joinKey* field is settable, but not readable. Using the *getConfig* command to query the ACL will produce a list of motes in the ACL, but for security reasons, the *joinKey* value is not returned in the query.

### Child Elements

Element	Type	Enum	Description	User Writable
macAddr	macAddress		MAC address of mote to be accepted into the network	Yes
joinKey	hex data		Join key of mote to be accepted into the network. Represented as 32-character hexadecimal string (representing 16 bytes)	Yes

## 5.5 Network

### Description

The *Network* element contains configuration elements for network-wide settings and statistics. Settings persist through Manager reset and power cycles, and require a system reset to take affect.

### Child Elements

Element	Type	Enum	Description	User Writable
netName	string		The network name is a user readable text field to describe the specific network installation	Yes
networkId	integer		Used to separate networks. Can be set during manufacturing, or in the field. HART recommends that some network IDs be reserved for specific purposes- see HCF Spec 75, table 2. 0xFFFF is never a valid network ID.	Yes
optimizationEnable	boolean		Enable network optimization.	Yes
maxMotes	integer		Maximum number of motes expected in the network.	Yes
numMotes	integer		Number of non-access point motes in Live state.	No
accessPointPA	integer		<i>(Updated in Manager 4.1)</i> The power setting for the Access Point power amplifier. Valid values are in the range -128 to 127.  In previous Manager versions, this is a boolean value where "True" means the Access Point power amplifier is enabled.  This parameter has no effect on products that do not have a power amplifier.	Yes
ccaEnabled	boolean		Enables or disables clear channel assessment for all motes in the network, including the access point mote.	Yes
requestedBasePkPeriod	integer		Defines the base bandwidth (in ms/packet) that the manager should allocate for each device.	Yes
minServicesPkPeriod	integer		Limits non-pipe services. Defines minimum data interval (in ms/packet) that a single mote may be allocated for the total of non-pipe, user requested bandwidth. Limits service requests from a mote.	Yes

minPipePkPeriod	integer		Limits bandwidth (ms/packet) on all pipes (both manager-API requested pipes and pipes as a result of service requests). This is most useful for regulating service requests from field devices.	Yes
bandwidthProfile	string	<a href="#">Bandwidth Profile</a>	The bandwidth profile determines the frame size (number of timeslots) for upstream, downstream, and advertising traffic.	Yes
manualUSFrameSize	integer		Manually sets the frame size (in number of timeslots) for upstream traffic. This parameter is only applicable if <i>bandwidthProfile</i> is set to <code>manual</code> .	Yes
manualDSFrameSize	integer		Manually sets the frame size (in number of timeslots) for downstream traffic. This parameter is only applicable if <i>bandwidthProfile</i> is set to <code>manual</code> .	Yes
manualAdvFrameSize	integer		Manually sets the frame size (in number of timeslots) for advertisement traffic. This parameter is only applicable if <i>bandwidthProfile</i> is set to <code>manual</code> .	Yes
maintStartTime	integer		Defines when to disable statistics collection. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.	Yes
maintEndTime	integer		Defines when to re-enable statistics collection. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.	Yes
netQueueSize	integer		Number of messages in the manager's network management queue.	No
userQueueSize	integer		Number of messages in the manager's user queue.	No
backboneEnabled	boolean		<i>Added in Manager 4.1.0.</i> Enables the network backbone.	Yes
backboneSize	integer		<i>Added in Manager 4.1.0.</i> Defines the size of the network backbone. Valid values are 4, 8, or 16 slots.	Yes
numParents	integer		<i>Added in Manager 4.1.0.</i> Defines the number of parents for each mote.	Yes
fastAdvPeriod	integer		<i>Added in Manager 4.1.0.</i> Duration of the fast advertisement period in milliseconds.	Yes
slowAdvPeriod	integer		<i>Added in Manager 4.1.0.</i> Duration of the slow advertisement period in milliseconds.	Yes

ChannelBlackList	---		Enables channel blacklisting of one or more channels for the entire network. When channels are blacklisted, they are not used by the network. See detailed description below. See the SmartMesh WirelessHART User's Guide section "Channel Blacklisting" for restrictions on the number of channels.	Yes
Sla	----		The service level agreement (SLA) allows the user to set service level agreements. When these limits are violated a corresponding alarm is sent via the notification channel. This element is described in the <a href="#">SLA section</a> .	Yes
OtapStatus	----		Status of remote software upgrade. This element is described in the <a href="#">OtapStatus section</a> .	No
Statistics	----		Returns a set of network statistics. This element is described in the <a href="#">Network Statistics section</a> .	No

This is an example of the Network element returned with depth "all".

```

<config><Network>
  <netName>myNet</netName>
  <networkId>182</networkId>
  <optimizationEnable>true</optimizationEnable>
  <maxMotes>251</maxMotes>
  <numMotes>0</numMotes>
  <accessPointPA>true</accessPointPA>
  <ccaEnabled>>false</ccaEnabled>
  <requestedBasePkPeriod>100000</requestedBasePkPeriod>
  <minServicesPkPeriod>400</minServicesPkPeriod>
  <minPipePkPeriod>480</minPipePkPeriod>
  <bandwidthProfile>P1</bandwidthProfile>
  <manualUSFrameSize>1024</manualUSFrameSize>
  <manualDSFrameSize>256</manualDSFrameSize>
  <manualAdvFrameSize>128</manualAdvFrameSize>
  <netQueueSize>0</netQueueSize>
  <userQueueSize>0</userQueueSize>
  <locationMode>off</locationMode>
  <ChannelBlackList>
    <frequency>2480</frequency>
  </ChannelBlackList>
  <Sla>
    <minNetReliability>99.00</minNetReliability>
    <maxNetLatency>12500</maxNetLatency>
    <minNetPathStability>50.00</minNetPathStability>
    <apRdntCoverageThreshold>70.00</apRdntCoverageThreshold>
  </Sla>
</Network></config>

```

The *ChannelBlacklist* contains a list of *frequency* elements that list the channels that are not used in the network. See Channel Blacklisting for a detailed description of the blacklist requirements.

Element	Type	Description	User writable
frequency	integer	The frequency that should not be used. Listed one or more times in the <i>ChannelBlacklist</i> element.	Yes

Querying the *ChannelBlacklist* returns the list of channels in the blacklist:

```

<config><Network>
  <ChannelBlackList>
    <frequency>2415</frequency>
    <frequency>2420</frequency>
    <frequency>2480</frequency>
  </ChannelBlackList>
</Network></config>

```

Setting a child element such as the *ChannelBlacklist* requires specifying the full container path:

Parameter name	Value
loginToken	my-login-token
configDocument	<config><Network><ChannelBlacklist><frequency>2805</frequency></Channe



## 5.6 Network Statistics

### Description

The *Network Statistics* element is a child of the *Network* element. It contains statistics accumulated over the whole network.

Statistics are accumulated in 15 minute intervals and can be queried over several ranges:

- lifetime - statistics accumulated over the lifetime of the network (since the last network reset).
- current - statistics accumulated in the current 15 minute interval
- short - statistics accumulated in a single 15 minute interval. One day (96 intervals) of 15 minute statistics are kept and can be queried by index – the latest full interval is index 0.
- long - statistics accumulated over one day. 7 days worth of daily statistics are kept and can be queried by index – the current day is index 0, yesterday is index 1, etc.

### Child Elements

Element	Type	Enum	Description	User Writable
netReliability	float		Reliability is the percentage of data packets generated that have been received. One-hundred percent reliability means that every data packet generated by a mote or given to it via serial API was received at the manager. The reported value is a network-wide average.	No
netPathStability	float		Path stability is the percentage of transmitted packets that have successfully reached their destination over a given path. Anything <100% path stability indicates that some packets were retried. The reported value is a network-wide average.	No
netLatency	integer		Latency is the average time in milliseconds required for a data packet to travel from the originating mote to the manager. Latency may vary across the network. The reported value is the network-wide average latency.	No
lostUpstreamPackets	integer		Total number of lost upstream packets across all devices and sessions. This field is only returned for lifetime statistics queries.	No

To retrieve lifetime Network Statistics, send the query:

```
<config><Network><Statistics><lifetime/></Statistics></Network></config>
```

To retrieve the current Network Statistics, send the query:

```
<config><Network><Statistics><statCur/></Statistics></Network></config>
```

To retrieve statistics from the previous day (long statistics interval), send:

```
<config><Network><Statistics>  
  <stat1DaySet><stat1Day><index>1</index></stat1Day></stat1DaySet>  
</Statistics></Network></config>
```


To retrieve statistics from the 15 minute interval (a short statistics interval) 30-45 minutes ago, send:

```
<config><Network><Statistics>  
  <stat15MinSet><stat15Min><index>2</index></stat15Min></stat15MinSet>  
</Statistics></Network></config>
```

## 5.7 Motes

### Description

Configuration of the network motes. The following elements apply to each mote (with the exception of the Access Point).

 The API allows querying and modifying user writable values for the Access Point, but these settings have no effect and do not persist when the Manager is restarted.

### Child Elements

Element	Type	Enum	Description	User Writable
macAddr	macAddress		Mote MAC address. The mote MAC address is a unique identifier of the mote.	No (identifier in queries)
moteld	integer		Numeric mote identifier. The <i>moteld</i> should not be used to uniquely identify a mote as it is not guaranteed to remain constant after a mote reset.	No
name	string		User-defined mote name	Yes
powerSource	string		Mote power source (line, battery, or rechargeable/power scavenging).	No
dischargeCurrent	integer		While the mote API uses discharge current, in $\mu\text{A}$ , the HART protocol uses peak packets/s (float) as the power limit for the device. This field contains $100 * \text{the peak packets/s}$ reported by the mote at join.  Note: The discharge current, discharge time, and recovery time are collectively used to describe the current sourcing capabilities of the three possible types of power supply feeding the mote.	No
dischargeTime	integer		The discharge time is the maximum time (in seconds) that the power supply can sustain the discharge current before experiencing a voltage drop.	No

recoveryTime	integer		The recovery time is the time (in seconds) required by the power supply to recover from a power drain (for example, recharge its capacitors).	No
enableRouting	boolean		Whether the mote may be configured for routing. (By default, motes are configured for routing.)	Yes
productName	string		Mote product name, <= 16 characters.	No
hwModel	integer		Mote hardware model	No
hwRev	integer		Mote hardware version	No
swRev	string		Mote firmware version. During mote startup, the version number is initially populated with the placeholder value "0.0.0-0" until the mote becomes operational and reports its value through the network (this may take a few minutes).	No
isAccessPoint	boolean		Whether this is an access point device	No
numJoins	integer		Total number of times the mote joined the network	No
state	string	<a href="#">Mote State</a>	Mote state	No
reason	string	<a href="#">Reason</a>	Indicates why the mote is in its current state.	No
joinTime	dustTime		Timestamp of mote join time. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.	No
voltage	float		The voltage is the reading from the last measurement of the mote power supply.	No
allocatedPkPeriod	integer		Currently allocated bandwidth (in ms/packet) from mote to gateway	No
allocatedPipePkPeriod	integer		Currently allocated bandwidth (in ms/packet) for a pipe	No
pipeStatus	string	<a href="#">Pipe Status</a>	Status of the pipe at the mote.	No
advertisingStatus	string	<a href="#">Advertising Status</a>	Status of advertising at the mote	No
numNeighbors	integer		The number of active neighbors used by the mote.	No

needNeighbor	boolean		Indicates whether a mote has only one parent and (or) insufficient links. A well formed network should have only one mote (the “single-parent” mote in the first hop) for which <i>needNeighbor</i> is <code>true</code> . If this parameter is true for any other mote, it indicates that an additional mote needs to be added nearby.	No
goodNeighbors	integer		Number of neighboring motes that have good (> 50%) quality paths	No
apRdntPeer	macAddress		MAC address of AP that shadows this one (only applicable for access points in redundant mode).	No
apRdntMode	string	AP Redundancy Mode	AP Redundancy mode (only applicable for access points in redundant system).	No
apRdntCoverage	float		AP Redundant coverage (only applicable for access points in redundant system).	No
locationTag	string		Reserved	No
maxNumNeighbors	integer		<i>Added in Manager 4.1.1.</i> Maximum number of neighbors that the mote can support. A value of 0 means unlimited.	No
numLinks	integer		<i>Added in Manager 4.1.1.</i> Number of currently allocated links on the mote.	No
maxNumLinks	integer		<i>Added in Manager 4.1.1.</i> Maximum number of links that the mote can support. A value of 0 means unlimited.	No
linksPerSec	float		<i>Added in Manager 4.1.1.</i> Current number of links per second allocated on the mote.	No
maxLinksPerSec	float		<i>Added in Manager 4.1.1.</i> Maximum number of links per second that the mote can support. A value of 0 means unlimited.	No
Statistics	---		Set of mote QOS statistics.	No

The *macAddr* is used as an identifier in queries. To retrieve a single mote, send the query:

```
<config><Motes><Mote><macAddr> 01-02-03-04-05-06-07-08 </macAddr></Mote></Motes></config>
```

## 5.8 Mote Statistics

### Description

The *Mote Statistics* element is a child of *Motes*. It contains a set of mote QOS statistics.

Statistics are accumulated in 15 minute intervals and can be queried over several ranges:

- lifetime - statistics accumulated over the lifetime of the network (since the last network reset).
- current - statistics accumulated in the current 15 minute interval
- short - statistics accumulated in a single 15 minute interval. One day (96 intervals) of 15 minute statistics are kept and can be queried by index – the latest full interval is index 0.
- long - statistics accumulated over one day. 7 days worth of daily statistics are kept and can be queried by index – the current day is index 0, yesterday is index 1, etc.

The voltage and temperature values are not accumulated across statistics periods.



### Access Point Statistics

Statistics for the AP should be ignored as they do not contain valid data.

### Child Elements

Element	Type	Enum	Description	User Writable
reliability	float		Reliability is the percentage of packets generated (internally or sent via API) by the mote that are received at the manager. One-hundred percent reliability means that every packet generated was received.	No
avgLatency	integer		Average elapsed time between timestamp on packet and receipt of packet by the manager in milliseconds.	No
numJoins	integer		Number of times mote joined the network in this statistic period	No
voltage	float		Volts remaining on mote battery	No
chargeConsumption	integer		Cumulative mC consumed by mote. Charge calculations are based on nominal part performance.	No
temperature	float		Temperature of mote (°C)	No
numLostPackets	integer		<i>Added in Manager 4.1.0.</i> Number of lost packets as calculated by the Manager based on the mote's security counter.	No

latencyToMote	integer		<i>Added in Manager 4.1.0.</i> Estimated downstream latency to the mote in milliseconds calculated based on the number of hops.	No
numDuplicates	integer		<i>Added in Manager 4.1.1.</i> Number of duplicate packets received by the Manager from the mote.	No

To retrieve lifetime statistics for the mote with MAC Address 01-02-03-04-05-06-07-08:

```
<config><Motes><Mote><macAddr>01-02-03-04-05-06-07-08</macAddr>
  <Statistics><lifetime/></Statistics>
</Mote></Motes></config>
```

To retrieve current statistics for the mote with MAC Address 01-02-03-04-05-06-07-08:

```
<config><Motes><Mote><macAddr>01-02-03-04-05-06-07-08</macAddr>
  <Statistics><statCur/></Statistics>
</Mote></Motes></config>
```

To retrieve statistics from the previous day for the mote with MAC Address 01-02-03-04-05-06-07-08:

```
<config><Motes><Mote><macAddr>01-02-03-04-05-06-07-08</macAddr>
  <Statistics>
    <stat1DaySet><stat1Day><index>1</index></stat1Day></stat1DaySet>
  </Statistics>
</Mote></Motes></config>
```

To retrieve statistics from the 15 minute interval starting 30-45 minutes ago for the mote with MAC Address 01-02-03-04-05-06-07-08:

```
<config><Motes><Mote><macAddr>01-02-03-04-05-06-07-08</macAddr>
  <Statistics>
    <stat15MinSet><stat15Min><index>2</index></stat15Min></stat15MinSet>
  </Statistics>
</Mote></Motes></config>
```

## 5.9 OtapStatus

### Description

The *OtapStatus* is a child element of [Network](#). It provides the status of a remote software upgrade.

### Child Elements

Element	Type	Enum	Description	User Writable
state	string	<a href="#">OTAP State</a>	Current OTAP state	No
elapsedTime	dustTime		Elapsed time since OTAP started. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.	No
totalDevices	integer		Total number of motes in the network	No
uploadedDevices	integer		Number of motes that have confirmed receipt of OTAP files	No
currentFile	string		Name of file currently being uploaded to motes	No
curFileTotalRetries	integer		Number of times to try sending current file	No
curFileCurrentRetry	integer		Current round of retrying current file	No
curFilePercentComplete	integer		Percent of fragments of current file that have been confirmed by all applicable motes	No
curFileSentFragments	integer		Number of fragments of current file that have been sent out (includes retries)	No
motes	---		Motes to be considered for upgrade. Child elements are described <a href="#">here</a> .	No



## 5.10 OtapStatus Mote

---

### Description

*Mote* is a child element of *OtapStatus*. It provides information about a mote to be considered for upgrade.

### Child Elements

Element	Type	Enum	Description	User Writable
moteld	integer		ID of mote	No
macAddr	macAddress		MAC address of mote	No
percentComplete	integer		Percent of fragments of applicable files that have been confirmed as received by the mote	No
status	string	<a href="#">Mote Otap Status</a>	Current mote OTAP status of this mote	No

## 5.11 Redundancy

---

### Description

Manager redundancy. (Reserved for future use.)

### Child Elements

Element	Type	Enum	Description	User Writable
localMode	string	<a href="#">Redundancy Mode</a>	Local redundancy mode	No
peerStatus	string	<a href="#">Redundancy Peer Status</a>	Redundancy peer status	No
peerControllerSwRev	string		Software version of the peer's manager ipackage	No

## 5.12 Security

### Description

Security mode and Access Control List (ACL). The security mode determines which motes the manager should accept into the network. The ACL contains the list of motes that have different join key from the common join key.

See Network Security for more information on how to use the ACL.

### Child Elements

Element	Type	Enum	Description	User Writable
securityMode	string	<a href="#">Security Mode</a>	Security mode of network. The security mode determines which motes the manager should accept into the network.	Yes
commonJoinKey	hex data		Network encryption password for joining with the common join key. Represented as 32-character hexadecimal string (representing 16 bytes)	Yes
acceptHARTDevicesOnly	boolean		If "True", the network only accepts devices with a HART prefix in the MAC address. address. By default this field is set to accept all devices.	Yes
Acl	----		Access Control List (ACL) is the list of motes that may be accepted into the network along with their join keys. Motes on the ACL do not use the common join key.  Child elements are described in the <a href="#">Acl</a> documentation.	----

## 5.13 Sla

### Description

*Sla* is a child element of the *Network* element. The Service Level Agreement (SLA) allows the user to configure network health parameters. When these limits are violated a corresponding alarm is sent via the notification channel.

### Child Elements

Element	Type	Enum	Description	User Writable
minNetReliability	float		Minimum allowable reliability (ratio of number of packets received at the manager to the number of packets expected at the manager) as percentage	Yes
maxNetLatency	integer		Maximum allowable packet latency (milliseconds)	Yes
minNetPathStability	float		Minimum allowable path stability (number of acknowledgments received/number of acknowledgments expected) as percentage	Yes
apRdntCoverageThreshold	float		Percentage threshold for access point redundant coverage (only used in redundant systems)	Yes

## 5.14 System

### Description

The *System* element contains general system information about the manager.

### Child Elements

Element	Type	Enum	Description	User Writable
systemName	string		Name of the network manager	Yes
location	string		Location of system	Yes
swRev	string		The version number applied to all software components of the Manager. The components include the Manager executable and other software that is distributed with the Manager device such as Admin Toolset and management scripts.	No
hwModel	string		Hardware model of the network manager	No
hwRev	string		Hardware version of the network manager	No
serialNumber	string		Serial number of the network manager	No
time	dustTime		Current time on the network manager. The time is represented as the number of milliseconds since 00:00:00.000 1/1/1970 GMT.	No
startTime	dustTime		Timestamp of the network manager startup. The time is represented as the number of milliseconds since 00:00:00.000 1/1/1970 GMT.	No
cliTimeout	integer		Time (in minutes) after which an inactive Manager CLI session ( <code>nwconsole</code> ) is dropped. If <code>cliTimeout</code> is set to 0, sessions are not dropped for inactivity.	Yes
controllerSwRev	string		The version of the Manager executable	No

An example of the *System* element retrieved through the *getConfig* command:

```
<config><System>
  <systemName>Test1</systemName>
  <location>Hayward</location>
  <swRev>4.0.0.21-6</swRev>
  <hwModel/>
  <hwRev/>
  <serialNumber>00170D80000D</serialNumber>
  <time>1334861097382</time>
  <startTime>1333583899000</startTime>
  <cliTimeout>1000</cliTimeout>
  <controllerSwRev>4.0.0.21</controllerSwRev>
</System></config>
```

## 5.15 Paths

### Description

The *Path* element describes a connection between two motes in the network.

### Child Elements

Element	Type	Enum	Description	User Writable
pathId	integer		Numeric path identifier	No
moteAMac	macAddress		The MAC address of a mote on one side of the path.	No
moteBMac	macAddress		The MAC address of a mote on the other side of the path.	No
numLinks	integer		A measure of how much bandwidth is allocated to this path. If <i>numLinks</i> is zero, this path is not used.	No
pathDirection	string	<a href="#">Path Direction</a>	Direction of communication between the two motes when sending to the manager. Downstream means mote B sends packets to mote A (B is downstream of A).	No
pathQuality	float		A measure of the quality of this path, where 100 is the best quality. Quality is the same as stability for a used path, and is based on RSSI for an unused path.	No
Statistics	---		A set of QOS statistics for a path. Child elements are described in <a href="#">Path Statistics</a> .	No

The Paths element can be used to query the list of neighbors of a particular mote. The following example shows the *getConfig* query for the neighbors of mote 00-17-0D-00-00-38-10-20:

```
<config><Paths><Path>
  <moteMac>00-17-0D-00-00-38-10-20</moteMac>
</Path></Paths></config>
```

The special field *moteMac* queries for all Paths that match the MAC address in either the *moteAMac* or *moteBMac* field.

If the specified mote has two neighbors, the response would contain:

```
<config><Paths>
  <Path>
    <pathId>65801</pathId>
    <moteAMac>00-17-0D-00-00-38-10-0A</moteAMac>
    <moteBMac>00-17-0D-00-00-38-10-20</moteBMac>
    <numLinks>2</numLinks>
    <pathDirection>upstream</pathDirection>
    <pathQuality>75</pathQuality>
  </Path><Path>
    <pathId>68553</pathId>
    <moteAMac>00-17-0D-00-00-38-10-20</moteAMac>
    <moteBMac>00-17-0D-00-00-38-10-EE</moteBMac>
    <numLinks>0</numLinks>
    <pathDirection>downstream</pathDirection>
    <pathQuality>75</pathQuality>
  </Path>
</Paths></config>
```



## 5.16 Path Statistics

### Description

*Path statistics* is a child of *Paths*. It contains a set of QOS statistics for a path.

Statistics are accumulated in 15 minute intervals and can be queried over several ranges:

- lifetime - statistics accumulated over the lifetime of the network (since the last network reset).
- current - statistics accumulated in the current 15 minute interval
- short - statistics accumulated in a single 15 minute interval. One day (96 intervals) of 15 minute statistics are kept and can be queried by index – the latest full interval is index 0.
- long - statistics accumulated over one day. 7 days worth of daily statistics are kept and can be queried by index – the current day is index 0, yesterday is index 1, etc.

### Child Elements

Element	Type	Enum	Description	User Writable
abPwr	integer		Average RSSI reading mote B has to neighbor mote A (dBm)	No
baPwr	integer		Average RSSI reading mote A has to neighbor mote B (dBm)	No
stability	float		Path stability (number of successful transmissions divided by total number of transmissions)	No

To retrieve lifetime statistics for the path with id 67890:

```
<config><Paths><Path><pathId>67890</pathId>
  <Statistics><lifetime/></Statistics>
</Path></Paths></config>
```

To retrieve current statistics for the path with id 67890:

```
<config><Paths><Path><pathId>67890</pathId>
  <Statistics><statCur/></Statistics>
</Path></Paths></config>
```

To retrieve statistics from the previous day for the path with id 67890:

```
<config><Paths><Path><pathId>67890</pathId>
  <Statistics>
    <stat1DaySet><stat1Day><index>1</index></stat1Day></stat1DaySet>
  </Statistics>
</Path></Paths></config>
```

To retrieve statistics from the 15 minute interval starting 30-45 minutes ago for the path with id 67890:

```
<config><Paths><Path><pathId>67890</pathId>
  <Statistics>
    <stat15MinSet><stat15Min><index>2</index></stat15Min></stat15MinSet>
  </Statistics>
</Path></Paths></config>
```

## 5.17 Users

---

### Description

The set of users with access to the Manager XML API. The user name and password are used to authenticate to the API in the [login command](#).

For security reasons, the password field is not returned in queries, but must be present when adding a user with *setConfig*.

### Child Elements

Element	Type	Enum	Description	User Writable
userName	string		User name	Yes
password	string		User password	Yes
privilege	string	<a href="#">User Privilege</a>	User privilege level. Users with <i>viewer</i> privilege can not perform configuration updates.	Yes

## 5.18 SourceRoute

### Description

(Added in Manager 4.1.0) The *SourceRoute* element contains a description of the primary and secondary downstream paths to a destination mote. The *SourceRoute* elements are read-only. The only query that is supported returns the *SourceRoute* for a single mote.

### Child Elements

Element	Type	Enum	Description	User Writable
destMacAddr	Mac Address		The MAC address of the destination mote	No (identifier in queries)
primaryPath	container element		The <i>primaryPath</i> element contains a list of <i>macAddr</i> elements that list the hops on the primary source route to the destination.	No
secondaryPath	container element		The <i>primaryPath</i> element contains a list of <i>macAddr</i> elements that list the hops on the secondary source route to the destination.	No

An example query document in the *getConfig* API call for a mote's SourceRoute:

```
<config><SourceRoutes><SourceRoute>
  <destMacAddr>00-17-0D-00-00-10-01-22</destMacAddr>
</SourceRoute></SourceRoutes></config>
```

An example response, in which there is no secondary path:

```
<config><SourceRoutes><SourceRoute>
  <destMacAddr>00-17-0D-00-00-10-01-22</destMacAddr>
  <primaryPath>
    <macAddr>00-17-0D-00-00-30-AA-BB</macAddr>
    <macAddr>00-17-0D-00-00-10-01-33</macAddr>
    <macAddr>00-17-0D-00-00-10-01-22</macAddr>
  </primaryPath>
  <secondaryPath/>
</SourceRoute></SourceRoutes></config>
```

## 5.19 Open Alarms

### Description

A [getConfig](#) query for the *Alarms* element retrieves a list of the open alarms on the Manager. Each *Alarm* element contains the timestamp, event identifier and alarm type information from the corresponding *alarmOpen* event. The [Alarms notification description](#) contains details about specific alarm types.

The only Alarm query that returns valid results is the query for all open alarms, see example below.

### Child Elements

Element	Type	Enum	Description	User Writable
timeStamp	integer		Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.	No
eventId	integer		Numeric identifier for the event.	No
<i>alarm type</i>	---		The specific alarm type that was opened. The element name is the alarm type and it may contain additional alarm-specific data.	No

### Examples

An example getConfig Query:

```
<config><Alarms/></config>
```

Response:

```
<config>
  <Alarms>
    <Alarm>
      <timeStamp>1379708963619</timeStamp>
      <eventId>1001</eventId>
      <moteDown>
        <macAddr>00-17-0D-00-00-01-02-03</macAddr>
      </moteDown>
    </Alarm>
  </Alarms>
</config>
```

## 6 Notifications

Notifications are pushed from the manager to a client over the Notification channel.

### 6.1 Data

#### Description

Motes may send upstream data to the manager. Each data packet generates a data notification. Unsolicited upstream data notifications are marked as responses. There are several optional fields that may be present in the data notification depending on the contents of the upstream data packet.

#### Notification Elements

Element	Type	Description
moteld	integer	Mote ID of the mote that generated the packet
macAddr	macAddress	MAC address of the mote that generated the packet
time	timestamp	Time when the mote <i>generated</i> the packet. The time is represented as the number of milliseconds since the Unix epoch, 00:00:00 Jan 1, 1970 UTC.
payload	hex data	Hex payload of the packet. The <code>type</code> attribute is deprecated and should be ignored.
isReliable	<i>(no data)</i>	Optional. If this element is present, the packet is reliable.
isRequest	<i>(no data)</i>	Optional. If this element is present, the packet is a request (and may require a response).
isBroadcast	<i>(no data)</i>	Optional. If this element is present, the packet is a broadcast packet.
callbackId	integer	Optional. Responses to reliable requests will contain the <code>callbackId</code> returned by the <i>sendRequest</i> call.
counter	integer	<i>Added in Manager 4.1.0.</i> Security counter of the data packet.

An unreliable upstream data packet:

```
<data>
  <moteId>6</moteId>
  <macAddr>00-1B-1E-00-00-00-04</macAddr>
  <time>1196989668944</time>
  <counter>1234</counter>
  <payload type='80'>0000030d03000006fc020174</payload>
</data>
```

A reliable response packet:

```
<data>
  <moteId>4</moteId>
  <macAddr>00-1B-1E-00-00-00-04</macAddr>
  <time>1351805476791</time>
  <isReliable/>
  <callbackId>31</callbackId>
  <counter>7711</counter>
  <payload type='80'>0000030d03000006fc020174</payload>
</data>
```

## 6.2 Events

System events describe system-level (often user-requested) changes.

Network events describe various events related to changes in the network topology. Unlike an [Alarm](#) event, which opens when a condition occurs and closes when the condition is no longer true, a network event is sent only once when the condition changes.

### 6.2.1 netPathCreate

#### Description

A connection (path) has been created between two motes in the network.

#### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPathCreate	----	Event name
pathId	integer	ID of the created path
moteAMac	macAddress	MAC address of the mote on one end of the path
moteBMac	macAddress	MAC address of the mote on the other end of the path

```

<event>
  <timeStamp>724295501</timeStamp>
  <eventId>319</eventId>
  <netPathCreate>
    <pathId>34</pathId>
    <moteAMac>00-00-00-00-00-00-00-01</moteAMac>
    <moteBMac>00-00-00-00-00-00-00-02</moteBMac>
  </netPathCreate>
</event>

```



## 6.2.2 netPathDelete

### Description

A connection (path) has been removed between two motes in the network.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPathDelete	----	Event name
pathId	integer	ID of the deleted path
moteAMac	macAddress	MAC address of a mote on one end of the path
moteBMac	macAddress	MAC address of the mote on the other end of the path

```

<event>
  <timeStamp>724330187</timeStamp>
  <eventId>320</eventId>
  <netPathDelete>
    <pathId>34</pathId>
    <moteAMac>00-00-00-00-00-00-01</moteAMac>
    <moteBMac>00-00-00-00-00-00-02</moteBMac>
  </netPathDelete>
</event>

```

## 6.2.3 netPathActivate

### Description

Communication was established or restored on a path. These notes have a link to send packets in an active frame.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPathActivate	----	Event name
pathId	integer	ID of the activated path
moteAMac	macAddress	MAC address of a mote on one end of the path
moteBMac	macAddress	MAC address of the mote on the other end of the path

```

<event>
  <timeStamp>724330187</timeStamp>
  <eventId>320</eventId>
  <netPathActivate>
    <pathId>34</pathId>
    <moteAMac>00-00-00-00-00-00-00-01</moteAMac>
    <moteBMac>00-00-00-00-00-00-00-02</moteBMac>
  </netPathActivate>
</event>

```

## 6.2.4 netPathDeactivate

### Description

A network path has been deactivated and is no longer used for active communication. The path will continue to exist until it is deleted. The path can be reused if it is activated again.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPathDeactivate	----	Event name
pathId	integer	ID of the path that was deactivated
moteAMac	macAddress	MAC address of a mote on one end of the path
moteBMac	macAddress	MAC address of the mote on the other end of the path

```

<event>
  <timeStamp>724330187</timeStamp>
  <eventId>320</eventId>
  <netPathDeactivate>
    <pathId>34</pathId>
    <moteAMac>00-00-00-00-00-00-00-01</moteAMac>
    <moteBMac>00-00-00-00-00-00-00-02</moteBMac>
  </netPathDeactivate>
</event>

```

## 6.2.5 netPathAlert

### Description

The manager received an alarm from a mote about this path indicating a mote did not receive a response from its peer within the allowable timeout.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPathAlert	----	Event name
pathId	integer	ID of the path for which the manager received an alarm.
moteAMac	macAddress	MAC address of a mote on one end of the path
moteBMac	macAddress	MAC address of the mote on the other end of the path

```

<event>
  <timeStamp>724337337</timeStamp>
  <eventId>323</eventId>
  <netPathAlert>
    <pathId>345</pathId>
    <macAddressA>00-00-00-00-00-00-00-01</macAddressA>
    <macAddressB>00-00-00-00-00-00-00-02</macAddressB>
  </netPathAlert>
</event>

```

## 6.2.6 netPipeOn

### Description

A pipe was turned on in the network as a result of a service request or a manager API command.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPipeOn	----	Event name
macAddr	macAddress	The MAC address of the pipe's endpoint mote
allocatedPipePkPeriod	integer	The usable bandwidth allocated to the pipe

```

<event>
  <timeStamp>724338782</timeStamp>
  <eventId>324</eventId>
  <netPipeOn>
    <macAddr>00-00-00-00-00-00-00-00</macAddr>
  </netPipeOn>
</event>

```

## 6.2.7 netPipeOff

### Description

A pipe was turned off in the network. The pipe's endpoint mote is identified by its MAC address.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPipeOff	----	Event name
macAddr	macAddress	MAC address of pipe destination

```

<event>
  <timeStamp>724341703</timeStamp>
  <eventId>325</eventId>
  <netPipeOff>
    <macAddr>00-00-00-00-00-00-00-00</macAddr>
  </netPipeOff>
</event>

```

## 6.2.8 netMoteJoin

### Description

A mote joined the network.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteJoin	---	Event name
moteId	integer	Mote ID of the joining mote
macAddr	macAddress	MAC address of the joining mote
reason	<a href="#">Reason</a>	Reason the mote is in the current state
userData	hex data	User data sent with the join request

```

<event>
  <timeStamp>724342931</timeStamp>
  <eventId>326</eventId>
  <netMoteJoin>
    <moteId>23</moteId>
    <macAddr>00-00-00-00-00-00-02</macAddr>
    <reason></reason>
    <userData>0000AF72B7C12094EE833326234</userData>
  </netMoteJoin>
</event>

```

## 6.2.9 netMoteJoinQuarantine

### Description

A mote has joined with a common key when quarantine mode is in use.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteJoinQuarantine	string	Event name
moteId	integer	Mote ID of the joining mote
macAddr	macAddress	MAC address of the joining mote
reason	Reason	Reason the mote is in the current state
userData	hex data	User data sent with the join request

```

<event>
  <timeStamp>724342931</timeStamp>
  <eventId>326</eventId>
  <netMoteJoinQuarantine>
    <moteId>23</moteId>
    <macAddr>00-00-00-00-00-00-02</macAddr>
    <reason></reason>
    <userData>0000AF72B7C12094EE833326234</userData>
  </netMoteJoinQuarantine>
</event>

```



## 6.2.10 netMoteLive

### Description

A mote in the network entered the **Live** state.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteLive	----	Event name
moteId	integer	ID of the mote
macAddr	macAddress	MAC address of the mote
reason	<a href="#">Reason</a>	Reason the mote is in the current state

```

<event>
  <timeStamp>724345543</timeStamp>
  <eventId>327</eventId>
  <netMoteLive>
    <moteId>9</moteId>
    <macAddr>00-00-00-00-00-00-00-07</macAddr>
    <reason></reason>
  </netMoteLive>
</event>

```

## 6.2.11 netMoteUnknown

### Description

A mote in the network has entered the **Unknown** state, which means the mote is no longer communicating in the network.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteUnknown	----	Event name
moteId	integer	ID of the mote that is <b>Unknown</b> (lost from the network)
macAddr	macAddress	MAC address of the mote that is <b>Unknown</b> (lost from the network)
reason	<a href="#">Reason</a>	Reason the mote is in the <b>Unknown</b> state

```

<event>
  <timeStamp>721801203</timeStamp>
  <eventId>272</eventId>
  <netMoteUnknown>
    <moteId>10</moteId>
    <macAddr>00-1B-1E-00-00-00-00-0A</macAddr>
    <reason/>
  </netMoteUnknown>
</event>

```

## 6.2.12 netMoteDisconnect

### Description

A mote in the network has entered the **Disconnecting** state, see [Mote State](#). This indicates that the mote is ready to be physically removed (traffic has been re-routed to make this mote a leaf node).

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteDisconnect	----	Event name
moteId	integer	Mote ID of the mote in the <b>Disconnecting</b> state
macAddr	macAddress	MAC address of the mote in the <b>Disconnecting</b> state
reason	<a href="#">Reason</a>	Reason the mote is in the <b>Disconnecting</b> state

```

<event>
  <timeStamp>721801203</timeStamp>
  <eventId>272</eventId>
  <netMoteDisconnect>
    <moteId>10</moteId>
    <macAddr>00-1B-1E-00-00-00-00-0A</macAddr>
    <reason/>
  </netMoteDisconnect>
</event>

```

## 6.2.13 netMoteJoinFailure

### Description

Mote failed to join the network (requires that the mote has sent in an initial join request - motes that never attempt to join are never in this state).

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteJoinFailure	----	Event name
macAddr	macAddress	MAC address of the mote that failed to join the network
reason	<a href="#">Reason</a>	Reason for the mote's join failure

```

<event>
  <timeStamp>724467949</timeStamp>
  <eventId>341</eventId>
  <netMoteJoinFailure>
    <macAddr>00-00-00-00-00-00-00-00</macAddr>
    <reason>JOINCNT</reason>
  </netMoteJoinFailure>
</event>

```

## 6.2.14 netMoteInvalidMIC

### Description

A packet received from this mote is invalid. An invalid MIC (message integrity check) may indicate a security problem. Each packet contains a MIC that allows the manager to verify the packet source and contents.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteInvalidMIC	----	Event name
macAddr	macAddress	MAC address of the mote whose packet failed the message integrity check (MIC).

```

<event>
  <timeStamp>724469176</timeStamp>
  <eventId>342</eventId>
  <netMoteInvalidMIC>
    <macAddr>00-00-00-00-00-00-00-06</macAddr>
  </netMoteInvalidMIC>
</event>

```

## 6.2.15 netPacketSent

### Description

When a *sendRequest* command is issued, a data packet is sent into the network and this notification is generated.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPacketSent	----	Event Name
callbackId	integer	The callback ID associated with the data packet that was injected into the network
macAddr	macAddress	The destination MAC address of the data packet that was injected into the network

```

<event>
  <timeStamp>724466480</timeStamp>
  <eventId>340</eventId>
  <netPacketSent>
    <callbackId>789</callbackId>
    <macAddr>00-00-00-00-00-00-00-08</macAddr>
  </netPacketSent>
</event>

```

## 6.2.16 netServiceDenied

### Description

The manager failed to allocate the requested service.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netServiceDenied	----	Event name
serviceId	integer	Service ID
requestingMacAddr	macAddress	MAC address of the mote that requested the service
peerMacAddr	macAddress	MAC address of the peer device for the service
appDomain	<a href="#">Application Domain</a>	Application domain for the requested the service
isSource	boolean	True if the mote requested this service as a source
isSink	boolean	True if the mote requested this service as a sink
isIntermittent	boolean	True if the mote requested this service as intermittent
period	integer	Requested service period (if the service is not intermittent) or latency (if the service is intermittent).

```
<event>
  <timeStamp>724470563</timeStamp>
  <eventId>343</eventId>
  <netServiceDenied>
    <serviceId>98</serviceId>
    <requestingMacAddr>00-00-00-00-00-00-00-03</requestingMacAddr>
    <peerMacAddr>00-00-00-00-00-00-00-07</peerMacAddr>
    <appDomain>publish</appDomain>
    <isSource>true</isSource>
    <isSink>false</isSink>
    <isIntermittent>false</isIntermittent>
    <period>567</period>
  </netServiceDenied>
</event>
```



## 6.2.17 netPingReply

### Description

This notification is generated as a result of pinging a mote in the network.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netPingReply	----	Event name
callbackId	integer	Callback ID associated with the ping.
macAddr	macAddress	MAC address of the mote that was pinged.
latency	integer	Round-trip time (in ms) for the ping.
temperature	float	Temperature (°C) reported by the mote.
voltage	float	Voltage (volts) reported by the mote.
hopCount	integer	Number of upstream hops the ping reply took.

```

<event>
  <timeStamp>724472285</timeStamp>
  <eventId>344</eventId>
  <netPingReply>
    <callbackId>70</callbackId>
    <macAddr>00-00-00-00-00-00-00-08</macAddr>
    <latency>0</latency>
    <temperature>22.50</temperature>
    <voltage>3.00</voltage>
    <hopCount>1</hopCount>
  </netPingReply>
</event>

```

## 6.2.18 sysBootUp

### Description

The manager started up.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysBootUp	----	Event name

```

<event>
  <timeStamp>721555130</timeStamp>
  <eventId>263</eventId>
  <sysBootUp/>
</event>

```

## 6.2.19 sysConnect

### Description

A user connected to the manager via one of the manager's interfaces.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysConnect	----	Event name
userName	string	Username used to connect to the system. Internal connections are indicated by the username <code>rpc-user</code>
ipAddr	string	IP address of the user who connected to the system
channel	<a href="#">Channel Type</a>	Interface on which the connection was made.

```

<event>
  <timeStamp>721695632</timeStamp>
  <eventId>266</eventId>
  <sysConnect>
    <userName>dust</userName>
    <ipAddr>192.168.1.110</ipAddr>
    <channel>cli</channel>
  </sysConnect>
</event>

```

## 6.2.20 sysDisconnect

### Description

User disconnected from the system. Internal notifications are indicated by the username `rpc-user`.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysDisconnect	----	Event name
userName	string	Name of user who disconnected from the system
ipAddr	string	IP address of the user who disconnected from the system
channel	<a href="#">Channel Type</a>	Interface from which the user disconnected

```

<event>
  <timeStamp>721759704</timeStamp>
  <eventId>268</eventId>
  <sysDisconnect>
    <userName>dust</userName>
    <ipAddr>192.168.1.110</ipAddr>
    <channel>cli</channel>
  </sysDisconnect>
</event>

```

## 6.2.21 sysManualMoteReset

### Description

This notification is sent when the manager sends a reset command to the mote. The notification does not indicate that the mote has received the reset command and actually reset itself. To determine the mote state, use the [getConfig](#) command to retrieve the Mote element. Note that this notification is not sent for unexpected mote resets, for example, due to power interruption or toggling the RST line on the mote

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysManualMoteReset	----	Event name
userName	string	Name of the user who reset the mote
moteId	integer	ID of the mote that was reset
macAddr	macAddress	MAC address of the mote that was reset

```

<event>
  <timeStamp>723793791</timeStamp>
  <eventId>312</eventId>
  <sysManualMoteReset>
    <userName>dustcli</userName>
    <moteId>7</moteId>
    <macAddr>00-00-00-00-00-00-07</macAddr>
  </sysManualMoteReset>
</event>

```

## 6.2.22 sysManualMoteDelete

### Description

This notification is sent when a user deletes a mote.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysManualMoteDelete	----	Event name
userName	string	Name of the user who deleted the mote
moteId	integer	ID of the mote that was deleted
macAddr	macAddress	MAC address of the mote that was deleted

```

<event>
  <timeStamp>721801169</timeStamp>
  <eventId>271</eventId>
  <sysManualMoteDelete>
    <userName>rpc-user</userName>
    <moteId>10</moteId>
    <macAddr>00-1B-1E-00-00-00-0A</macAddr>
  </sysManualMoteDelete>
</event>

```

## 6.2.23 sysManualMoteDecommission

### Description

This notification is sent when the manager starts to decommission a mote so it can be removed from the network. The notification does not indicate that the mote has been disconnected. To determine the mote state, use the [getConfig](#) command to retrieve the Mote element.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for event
sysManualMoteDecommission	----	Event name
userName	string	Name of the user who decommissioned the mote
moteId	integer	Mote ID of the mote that was decommissioned
macAddr	macAddress	MAC address of the mote that was decommissioned

```

<event>
  <eventId>1234</eventId>
  <time>9999</time>
  <sysManualMoteDecommission>
    <userName>dustuser</userName>
    <moteId>23</moteId>
    <macAddr>00-00-00-00-00-00-02</macAddr>
  </sysManualMoteDecommission>
</event>

```

## 6.2.24 sysManualNetReset

### Description

This notification is sent when the manager receives a command to reset the network.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT
eventId	integer	Numeric identifier for the event
sysManualNetReset	----	Event name
userName	string	Name of user who reset the network

```

<event>
  <timeStamp>723902311</timeStamp>
  <eventId>315</eventId>
  <sysManualNetReset>
    <userName>admin</userName>
  </sysManualNetReset>
</event>

```



## 6.2.25 sysManualDccReset

### Description

This notification is sent when a user resets the manager by sending the *reset* command.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT,
eventId	integer	Numeric identifier for the event
sysManualDccReset	----	Event name
userName	string	Name of the user who reset the manager

```

<event>
  <timeStamp>723937539</timeStamp>
  <eventId>316</eventId>
  <sysManualDccReset>
    <userName>admin</userName>
  </sysManualDccReset>
</event>

```

## 6.2.26 sysManualStatReset

### Description

This notification is sent when the manager receives a command to clear its internal statistics collection.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysManualStatReset	----	Event name
userName	string	Name of the user who reset all statistics

```

<event>
  <timeStamp>723985707</timeStamp>
  <eventId>317</eventId>
  <sysManualStatReset>
    <userName>admin</userName>
  </sysManualStatReset>
</event>

```

## 6.2.27 sysConfigChange

### Description

This notification is sent when a user has changed a configuration element. This notification is most useful when multiple clients are simultaneously connected to the manager. The config change notification contains the following information:

- User that generated the change
- Type of object that was changed
- ID of the object that was changed

The following object types are reported:

- mote
- path
- network
- SLA
- system
- user
- blacklist
- security
- ACL

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysConfigChange	----	Event name
userName	string	Name of the user who changed the configuration object
objectType	string	Type of configuration object that was changed (see list above)
objectId	string	Identifier of the object that was changed (if appropriate)

```
<event>
  <timeStamp>1198179363114</timeStamp>
  <eventId>106484</eventId>
  <sysConfigChange>
    <userName>admin</userName>
    <objectType>system</objectType>
    <objectId/>
  </sysConfigChange>
</event>
```

## 6.2.28 sysCmdFinish

### Description

This notification is sent when a command associated with a configuration change finishes executing. The notification contains the following information:

- Callback ID associated with the command
- Type of object that was changed
- ID of the object that was changed
- MAC address of the mote (if a mote is associated with the command)
- Command's result code

The following object types are reported:

- mote
- path
- network
- networkId
- networkKey
- joinKey
- sessionKey
- sla
- system
- user
- channelBlackList
- security
- acl
- redundancy

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysCmdFinish	----	Event name
callbackId	integer	Callback ID associated with the command
objectType	<a href="#">Object Type</a>	Type of command that finished executing

---

macAddr	macAddress	MAC address of the mote (if the command is associated with a mote).
resultCode	integer	Code indicating the command result

## 6.2.29 sysRdntModeChange

### Description

This notification is sent when the redundancy mode has changed.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysRdntModeChange	----	Event name
redundancyMode	<a href="#">Redundancy Mode</a>	New redundancy mode
redundancyModeReason	<a href="#">Redundancy Mode Reason</a>	Reason for the new redundancy mode

```

<event>
  <timeStamp>724356578</timeStamp>
  <eventId>332</eventId>
  <sysRdntModeChange>
    <redundancyMode>transToMaster</redundancyMode>
    <redundancyModeReason>manual</redundancyModeReason>
  </sysRdntModeChange>
</event>

```

## 6.2.30 sysRdntPeerStatusChange

### Description

This notification is sent when the redundancy peer status has changed.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
sysRdntPeerStatusChange	----	Event name
redundancyPeerStatus	<a href="#">Redundancy Peer Status</a>	New peer status

```

<event>
  <timeStamp>724356578</timeStamp>
  <eventId>332</eventId>
  <sysRdntPeerStatusChange>
    <redundancyPeerStatus>connected</redundancyPeerStatus>
  </sysRdntPeerStatusChange>
</event>

```



## 6.2.31 netReset

### Description

The network was reset. All nodes must rejoin.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netReset	----	Event name

```

<event>
  <timeStamp>721555138</timeStamp>
  <eventId>745</eventId>
  <netReset/>
</event>

```

## 6.2.32 netTransportTimeout

### Description

(Added in Manager 4.1.0) The `netTransportTimeout` event indicates that a timeout has occurred on the gateway transport session.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netTransportTimeout	----	Event name
srcMacAddr	macAddress	Source address for the request that timed out. In version 4.1.0, the source address is always the gateway MAC address, i.e. 00-1B-1E-F9-81-00-00-02.
destMacAddr	macAddress	Destination address for the request that timed out.
timeoutType	Timeout Type	Description of the timeout.
callbackId	integer	Callback ID associated with the request that timed out.

```

<event>
  <timeStamp>123456999</timeStamp>
  <eventId>42</eventId>
  <netTransportTimeout>
    <srcMacAddr>00-1B-1E-F9-81-00-00-02.</srcMacAddr>
    <destMacAddr>00-1B-1E-AB-CD-00-00-01</destMacAddr>
    <timeoutType>Retry</timeoutType>
    <callbackId>17</callbackId>
  </netTransportTimeout>
</event>

```

## 6.2.33 netMoteQuarantine

### Description

A mote in the network entered the **Quarantine** state.

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the event
netMoteQuarantine	----	Event name
moteId	integer	ID of the mote
macAddr	macAddress	MAC address of the mote
reason	Reason	Reason the mote is in the current state

```

<event>
  <timeStamp>724345543</timeStamp>
  <eventId>377</eventId>
  <netMoteQuarantine>
    <moteId>7</moteId>
    <macAddr>00-00-00-00-00-00-14</macAddr>
    <reason/>
  </netMoteQuarantine>
</event>

```

## 6.3 Alarms

### Description

An alarm notification indicates a persistent network condition of a specific type described within the alarm. An *alarmOpen* notification is generated when the manager detects the alarm condition. An *alarmClosed* notification is generated when the manager detects that the alarm condition is no longer present.

### Notification Elements

Element	Type	Description
moteDown	string	This alarm is generated when a mote fails to respond to message from the manager. The <i>moteDown</i> element contains a <i>macAddr</i> element with the MAC address of the problematic mote.
slaReliability	(no data)	This alarm is generated when network reliability drops below the preset SLA threshold.
slaLatency	(no data)	This alarm is generated when network latency goes above the preset SLA threshold.
slaStability	(no data)	This alarm is generated when network stability drops below the preset SLA threshold.
maxMotesReached	(no data)	This alarm is generated when the network reaches the maximum number of motes for which the network has been configured.
bbLatencyWarn	(no data)	(Added in Manager 4.1) This alarm is generated when the network contains DN2510-based motes and the upstream backbone is used. DN2510 motes cannot participate in the backbone, so latency targets may not be met.

## 6.3.1 alarmOpen

### Description

An alarm was opened in response to a network event. For alarm details, see [Alarms](#).

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the event was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for the <i>alarmOpen</i> event.
alarmOpen	----	Details of the alarm event. The contents of the <i>alarmOpen</i> element indicate the type of alarm and contain details related to the alarm. For a list of alarms, see <a href="#">Alarms</a> .

```

<event>
  <timeStamp>721801084</timeStamp>
  <eventId>270</eventId>
  <alarmOpen>
    <moteDown>
      <macAddr>00-1B-1E-00-00-00-00-0A</macAddr>
    </moteDown>
  </alarmOpen>
</event>

```

## 6.3.2 alarmClose

### Description

This event is generated when an alarm condition is corrected. For list of alarms, see [Alarms](#).

### Notification Elements

Element	Type	Description
timeStamp	timestamp	Time the notification was generated. The time is represented as the number of milliseconds since 00:00:00:000 1/1/1970 GMT.
eventId	integer	Numeric identifier for an event
alarmClose	----	Name of the alarm. For list of alarms, see <a href="#">Alarms</a> .
alarmOpenEventId	integer	ID of the event that opened the alarm

```

<event>
  <timeStamp>721801204</timeStamp>
  <eventId>273</eventId>
  <alarmClose>
    <moteDown>
      <macAddr>00-1B-1E-00-00-00-0A</macAddr>
    </moteDown>
    <alarmOpenEventId>270</alarmOpenEventId>
  </alarmClose>
</event>

```

## 6.4 Log and Error Messages

---

### Description

Log and error messages are used for network troubleshooting. These messages are written to a rotating log file on the manager.

### Notification Elements

Element	Type	Description
time	timestamp	Time the message was generated. The time is represented as the number of milliseconds since 00:00:00.000 1/1/1970 GMT.
severity	integer	Severity of the message. Values are 0 (debug), 1 (trace), 2 (informational), 3 (warning), 4 (error), 5 (fatal error)
message	string	Log message

## 6.5 CLI Notifications

---

### Description

*CLI notifications are deprecated and should not be used.* This notification is generally sent in response to *cli* commands, however some CLI notifications are sent unsolicited, e.g. certain security errors. Each CLI notification contains one line of the CLI output from a command.

### Notification Elements

Element	Type	Description
time	timestamp	Time output was generated
message	string	One line of CLI output. Output is sent line by line. Each line has a maximum of 80 characters. Lines longer than 80 characters are split into multiple notifications.
endOfResponse	<i>(no data)</i>	If the <i>endOfResponse</i> element is present, this notification contains the last line of the CLI command's output.



## 6.6 Health Reports

---

### Description

(Added in Manager 4.1.0) Health Report notifications are generated when the Manager receives one of several HART commands from a mote in the network. The notification contains the raw payload of one or more HART commands.

The following commands may be contained in a `stdMoteReport` element (see HCF Spec-155 for format of HART commands):

- 779 (0x030B) - Device Health Report
- 780 (0x030C) - Neighbor Health Report
- 787 (0x0313) - Neighbor Signal Levels

The following commands may be contained in a `vendorMoteReport` element:

- 64515 (0xFC03) - Dust Device Health Report

DN2510 and Eterna-based motes will return the following fields:

- `uint8_t length` // 0x0E (14 bytes)
- `uint8_t response code` // 0x00 (always 0)
- `uint16_t extDevCode`; // Device code 0xE0A2
- `uint32_t chargeConsumption`; // Lifetime charge consumption (mC)
- `uint8_t meanQueueOcc:4`; // Mean queue occupancy
- `uint8_t maxQueueOcc:4`; // Max queue occupancy
- `uint16_t pktsForwarded`; // Number of packets forwarded
- `uint8_t pktsDropped`; // Number of packets dropped
- `int8_t temperature`; // Temperature (°C)
- `uint16_t voltage`; // Battery voltage (mV)

Eterna-based motes may (depending upon manager version) return a length of 0x12 (18 bytes) and return the following additional fields:


- `uint16_t numAppTxAttempts`; // Number of times the application tried to hand a packet to the mote
- `uint16_t numAppTxFails`; // Number of times the application was NACKed
- 64549 (0xFC25) - Dust RSSI Report (available in mote >= 1.1.0)
- `uint8_t length` // 0x4D (77 bytes)
- `uint8_t response code` // 0x00 (always 0)
- `uint16_t extDevCode`; // Device code 0xE0A2

- `rss_report_t report[15]; // Array of RSSI reports for each channel`

Where each RSSI report consists of:

- `int8_t avgIdleRSSI; // Average RSSI measured during idle listens`
- `uint16_t txUnicastCnt; // Number of unicast attempts on this channel`
- `uint16_t txUnicastFailCnt; // Number of failures on this channel`

The Dust RSSI report contains measurements of background radio energy and per-channel path stability during mote idle receive timeslots. It can be used to find localized in-band interferers that may be affecting network performance.

 In general, a client application should expect that health reports (and other notifications) may present additional fields in later versions. The client should use the length byte to parse the payload, discarding unknown new fields as appropriate.

## Notification Elements

Element	Type	Description
macAddr	macAddress	MAC address of the mote that generated the HART command.
time	timestamp	Time when the mote <i>generated</i> the command. The time is represented as the number of milliseconds since the Unix epoch, 00:00:00 Jan 1, 1970 UTC.
payload	hex data	The HART command as hex data.

An example notification for HART command 779 (Device Health Report):

```
<stdMoteReport>  
  <macAddr>00-17-0D-00-00-10-13-1D</macAddr>  
  <time>2773166646</time>  
  <payload>030c0e00000101000101b901070064001b</payload>  
</stdMoteReport>
```

## 7 Definitions

---

### 7.1 Advertising Status

---

Value	Description
on	Advertising is on
off	Advertising is off
pending	Manager is in the process of changing advertising from on to off (or vice-versa)

### 7.2 AP Redundancy Mode

---

Value	Description
standalone	AP is operating in standalone mode (not operating as a master or a slave)
transToMaster	AP is transitioning to master mode
transToSlave	AP is transitioning to slave mode
master	AP is operating in master mode
slave	AP is operating in slave mode
failed	AP has failed and has not recovered

### 7.3 Application Domain

---

The application domain field specifies the type of packet to be sent to the mote.

Value	Description
publish	Data packets sent on a regular, periodic basis
event	Packets (such as alarms) that are triggered by an event
maintenance	A series of request/response packets used to manage the device

## 7.4 Bandwidth Profile

The bandwidth profile defines the default bandwidth available when a mote is connected to the network.

Value	Description
P1	Normal profile
P2	Low-power profile
Manual	Manual profile, which allows you to set the number of timeslots for upstream, downstream, and advertising traffic. The manual bandwidth profile is an advanced feature to be used only with direct support.

## 7.5 Channel Type

The channel type describes the Manager interface to which a connection was made.

Value	Description
cli	Manager command line
config	Manager XML API control channel
notif	Manager XML API notification channel

## 7.6 Response Codes

The following response codes may be returned by a command.

Error Code	Error Name	Description
0	API_OK	The application layer has processed the command successfully
1	API_END_OF_LIST	The iteration has reached the end of the list of objects
2	API_OBJECT_NOT_FOUND	The object was not found
3	API_VAL_CROSSREF	Validation error. The cross reference failed.
4	API_VAL_ENUM	Validation error. The value was not in the enumeration.
5	API_VAL_PATTERN	Validation error. The value contained one of the following invalid characters: @, &, <, or >.

6	API_VAL_PATTERN_HEX	Validation error. The value contained an invalid hexadecimal character (valid characters are a-f, A-F, and 0-9).
7	API_VAL_ENCKEY	Validation error. The encryption key must contain exactly 32 hex characters.
8	API_VAL_DATAPACKET	Validation error. The message does not fit in the packet.
9	API_VAL_REQFIELD	Validation error. A required field is missing.
10	API_VAL_LISTLEN	Validation error. Cannot add the new object because the limit has been reached.
11	API_VAL_UNCHANGEABLE	Validation error. The element cannot be changed.
12	API_NOTUNIQCMD	A similar command is already being processed
13	API_VAL_STATE	Validation error. The current state does not allow this action to be performed.
14	API_VAL_MINMAX	Validation error. The maximum value must be greater than the minimum value.
15	API_VAL_TIMEBOUNDARY	Validation error. The time is not on the required 15-minute boundary.
16	API_VAL_STARTEND	Validation error. The end time must be after start time.
17	API_VAL_LAST_ELEMENT	Validation error. Cannot delete the last element.
18	API_NETLAYER_FULL	The manager's transmit queue is full
19	API_VAL_CANT_CREATE	Validation error. Cannot create the object because it already exists.
20	API_CLI_NULL_COMMAND	Empty CLI command
21	API_CLI_WRITE_ERROR	Could not write to the CLI session
22	API_INVALID_MOTE	The specified mote is invalid
23	API_INVALID_COMMAND	The command is invalid
24	API_INVALID_ARGUMENT	The argument to the command is invalid
27	API_INVALID_AUTHENTICATION	The command could not be authenticated
28	API_SLAVE	(Reserved for future use.)
29	API_LICENSE	Invalid license key
30	API_NO_ACL_ENTRY	The ACL entry doesn't exist
31	API_SERVICE_DENIED	The service was denied
232	API_GENERAL_ERROR	A general error that does not fall into any other error category

1001	XML_INVALID_FORMAT	The XML format is invalid
1002	XML_INVALID_DATA	The XML data is invalid
1003	XML_PARSE_ERROR	A parsing error occurred
1004	XML_INVALID_RPC_CONNECTION	Internal error
1005	XML_INVALID_AUTHENTICATION	The command could not be authenticated
1006	XML_INVALID_NOTIF_CLIENT	The notification client does not exist
1007	XML_TOO_MANY_NOTIF_CLIENTS	The maximum number of notification clients has been reached

## 7.7 Frame State

Indicates the current state of the frame.

Value	Description
on	Frame is on
off	Frame is off
activating	Frame is being activated
deactivating	Frame is being deactivated

## 7.8 Mote OTAP Status

---

Value	Description
NotInOtap	Mote is not receiving OTAP at this moment because the OTAP file does not apply to this mote.
InProgress	OTAP is in progress on this mote
Cancelled	OTAP was cancelled for this mote
LockedOut	OTAP will not be performed on this mote because OTAP lockout was set for this mote
LowBattery	Mote battery power is too low for OTAP to be performed
NoSpace	Unexpected error. There is not enough space in the mote flash for the OTAP file.
FileError	Unexpected error. The OTAP file is invalid or cannot be recognized by the mote.
MicError	Unexpected internal error occurred
Finished	OTAP was completed on this mote

## 7.9 Mote State

---

The mote state

Value	Description
idle	Mote has not been part of the network since the manager started
lost	Mote is not currently part of the network
negotiating1	Mote is in the process of joining the network
negotiating2	Mote is in the process of joining the network
connected	Mote is connected to the network
operational	Mote is operational
disconnecting	Mote may be physically removed from network without affecting network



## 7.10 Object Type

---

Value	Description
mote	Mote
path	Path
network	Network
SLA	Service level agreement
system	System
user	User
blacklist	Channel blacklist
security	Security settings
ACL	Access control list
networkId	Network ID
networkKey	Network key
joinKey	Join key
sessionKey	Session key
redundancy	Redundancy settings

## 7.11 OTAP State

---

Value	Description
Idle	OTAP process has been cancelled
Running	OTAP handshakes are being performed in preparation for uploading files to the motes
Uploading	OTAP files are being uploaded to the motes
Committing	Upload is now complete and motes are being reset in order to activate the new software. Cancelling an OTAP at this stage will result in motes running different software versions
Completed	OTAP is finished

## 7.12 Packet Priority

---

The priority field specifies the packet priority.

Value	Description
high	Priority is high
medium	Priority is medium
low	Priority is low

## 7.13 Path Direction

---

The direction of a path is defined with respect to motes A and B, which are identified in the Path element. The direction is relative to the manager.

Value	Description
all	Select both upstream and downstream paths.
upstream	Upstream means mote B is a child of mote A (B is upstream of A)
downstream	Downstream means mote B is a child of mote A (B is downstream of A)
unused	An unused path is not currently used for communication between motes A and B.

## 7.14 Pipe Direction

---

The pipe direction field is case sensitive.

Value	Description
UniUp	Upstream
UniDown	Downstream
Bi	Both upstream and downstream

## 7.15 Pipe Status

---

Value	Description
-------	-------------

off	Pipe is off
pending	Manager is in the process of changing the state of the pipe from on to off, or vice-versa)
on_bi	Pipe is on for bidirectional traffic
on_up	Pipe is on for upstream traffic
on_down	Pipe is on for downstream traffic

## 7.16 Reason

The reason field describes why the mote is in its current state.

Value	Description
MAXMOTES	Maximum number of motes was reached
UNREACH	Unreachable
NOTCONN	Not connected
CFGERR	Configuration error
MICERR	MIC error
JOINCNT	Invalid join counter
NOACL	No ACL entry for this mote
JOINTIMEOUT	Timeout during join messages
SWITCHOVER	Join dropped during switchover
COMPATIBILITY	AP is not compatible

## 7.17 Redundancy Peer Status

The redundancy peer status describes the state of the other member of the redundancy pair.

Value	Description
unknown	State of the peer manager is unknown (not connected)
connected	Peer manager is connected, but the state is not synchronized between the managers
synchronized	State of the peer manager is synchronized with the local state, indicating that it is possible to failover

## 7.18 Redundancy Mode

---

Value	Description
standalone	Manager is operating in standalone mode (not operating as a master or a slave)
transToMaster	Manager is transitioning to master mode
transToSlave	Manager is transitioning to slave mode
master	Manager is operating in master mode
slave	Manager is operating in slave mode
failed	Manager has failed and has not recovered

## 7.19 Redundancy Mode Reason

---

The reason field describes why the manager for current redundancy mode.

Value	Description
promote	The manager was promoted to be the master.
manual	The manager redundancy mode was updated manually.
apfailed	The manager was demoted because of an AP failure.
localFailure	The manager was demoted because of a system or communication failure.

## 7.20 Reset Object

---

Value	Description
system	Resets manager software
network	Resets the network
stat	Resets the manager's collection of statistics
eventLog	Resets the manager's log of events
mote	Resets a mote

## 7.21 Security Mode

---

The security mode defines whether the manager allows motes to join with the common join key.

Value	Description
acceptACL	When the security mode is set to accept ACL, only motes on the access control list (ACL) may be accepted into the network.
acceptCommonJoinKey	When the security mode set to accept the common join key, motes may be accepted into the network if they have the common join key or if they are listed on the ACL.
quarantineOnCommonJoinKey	When a mote joins with the common join key, it is placed in a quarantine state.

## 7.22 Timeout Type

---

The timeout type describes what timeout was associated with a *netTransportTimeout* event.

Value	Description
retry	The manager did not receive a response and will retry sending the packet.


## 7.23 User Privilege

---

User privilege describes the level of access that a user has to the manager API.

Value	Description
viewer	Users with the viewer privilege can view, but can not change manager configuration.
user	Users with the user privilege can view and update manager configuration.

## 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

This documentation is protected by United States and international copyright and other intellectual and industrial property laws. It is solely owned by Linear Technology and its licensors and is distributed under a restrictive license. This product, or any portion thereof, may not be used, copied, modified, reverse assembled, reverse compiled, reverse engineered, distributed, or redistributed in any form by any means without the prior written authorization of Linear Technology.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a), and any and all similar and successor legislation and regulation.

## 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.

Linear Technology does not warrant or represent that any license, either express or implied, is granted under any Linear Technology patent right, copyright, mask work right, or other Linear Technology intellectual property right relating to any combination, machine, or process in which Linear Technology products or services are used. Information published by Linear Technology regarding third-party products or services does not constitute a license from Linear Technology to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Linear Technology under the patents or other intellectual property of Linear Technology.

Dust Networks, Inc is a wholly owned subsidiary of Linear Technology Corporation.

© Linear Technology Corp. 2012-2015 All Rights Reserved.