

# MCUmgr Bluetooth protocol (SMP)

- [About](#)
- [SMP Service](#)
- [Simple Management Protocol](#)
  - [SMP Header](#)
  - [Payload](#)
- [Commands](#)
- [Device Firmware Upgrade](#)
  - [Modes](#)
    - [Test and Confirm](#)
    - [Test only](#)
    - [Confirm only](#)
  - [Algorithm](#)
- [Mobile clients](#)

## About

Implementation of mcumgr may be found here: <https://github.com/apache/mynewt-mcumgr>

Supported are Mynewt and Zephyr.

## SMP Service

SMP stands for Simple Management Protocol.

The GATT SMP service details:

Overview	Characteristics
----------	-----------------

# Simple Management Protocol

Each SMP packet consists of 8-byte header and optional payload.

The server must respond to each SMP packet with a notification.

## SMP Header

Bytes	Comment	Possible Values								
0	<p>Operation.</p> <p>An outgoing command will either be a READ or a WRITE which will trigger a READ RSP or WRITE RSP from the end device.</p> <p>Big Endian</p> <p><b>Links:</b></p> <p><a href="https://github.com/apache/mynewt-mcumgr/blob/master/mgmt/include/mgmt/mgmt.h">https://github.com/apache/mynewt-mcumgr/blob/master/mgmt/include/mgmt/mgmt.h</a></p>	<table><tr><td>READ</td><td>0</td></tr><tr><td>READ RSP</td><td>1</td></tr><tr><td>WRITE</td><td>2</td></tr><tr><td>WRITE RSP</td><td>3</td></tr></table>	READ	0	READ RSP	1	WRITE	2	WRITE RSP	3
READ	0									
READ RSP	1									
WRITE	2									
WRITE RSP	3									
1	<p>Optional flags fro this command.</p> <p>Not used. Shall be 0.</p>	N/A								
2-3	<p>Payload length, in bytes.</p> <p>UINT16, Big Endian.</p>									

4-5	<p>Group ID.</p> <p>The Command ID (see below) references the subcommand of this group. Group IDs up to unsigned integer 64 are reserved for system level McuMgr commands. Per-user commands can be implemented for Group IDs above 64.</p> <p>UINT16, Big Endian.</p> <p><b>Links:</b></p> <p><a href="https://github.com/apache/mynewt-mcumgr/blob/master/mgmt/include/mgmt/mgmt.h">https://github.com/apache/mynewt-mcumgr/blob/master/mgmt/include/mgmt/mgmt.h</a> - group IDs</p> <p><a href="https://github.com/apache/mynewt-mcumgr/tree/master/cmd">https://github.com/apache/mynewt-mcumgr/tree/master/cmd</a> - groups implementation folder</p>	<table><tr><td>OS*</td><td>0</td></tr><tr><td>IMAGE*</td><td>1</td></tr><tr><td>STAT*</td><td>2</td></tr><tr><td>CONFIG</td><td>3</td></tr><tr><td>LOG</td><td>4</td></tr><tr><td>CRASH</td><td>5</td></tr><tr><td>SPLIT</td><td>6</td></tr><tr><td>RUN</td><td>7</td></tr><tr><td>FS*</td><td>8</td></tr><tr><td>PER USER</td><td>64</td></tr></table> <p>* Partially or fully implemented in Zephyr</p>	OS*	0	IMAGE*	1	STAT*	2	CONFIG	3	LOG	4	CRASH	5	SPLIT	6	RUN	7	FS*	8	PER USER	64
OS*	0																					
IMAGE*	1																					
STAT*	2																					
CONFIG	3																					
LOG	4																					
CRASH	5																					
SPLIT	6																					
RUN	7																					
FS*	8																					
PER USER	64																					
6	<p>Sequence Number.</p> <p>The sequence number in the request shall match sequence number in the response. This allows, in theory, to sent multiple requests one after each other without waiting for the response, and then receiving responses with matching sequence numbers. However, this does not guarantee that the protocol sent using this mechanism will support it, e.g. DFU in NCS 1.7 seems to be sending wrong responses (incorrect offset) even though the sequence numbers are returned correctly, making the DFU much slower due to unnecessary repetitions. In Android library this may be utilized using <a href="https://github.com/NordicSemiconductor/Android-nRF-Connect-Device-Manager/blob/a61a0294fb4241ffc4c91621c1a8ede3d4c4643/mcumgr-core/src/main/java/io/runtime/mcumgr/dfu/FirmwareUpgradeManager.java#L308">https://github.com/NordicSemiconductor/Android-nRF-Connect-Device-Manager/blob/a61a0294fb4241ffc4c91621c1a8ede3d4c4643/mcumgr-core/src/main/java/io/runtime/mcumgr/dfu/FirmwareUpgradeManager.java#L308</a></p>	N/A																				
7	<p>Command ID.</p> <p>The Command ID identifies the subcommand for the group set by Group ID field.</p> <p>Command IDs may be found in <a href="https://github.com/apache/mynewt-mcumgr/tree/master/cmd">https://github.com/apache/mynewt-mcumgr/tree/master/cmd</a> Group include [group]_mgmt.h</p>	See below.																				

## Payload

Payload shall be encoded using CBOR protocol. CBOR is a JSON but for embedded.

Link: <http://cbor.io>

CBOR 2 JSON converter: <http://cbor.me>

Example:

JSON:

```
{ "len": 11, "offset": 0, "data": "hello world" }
```

CBOR:

```
A3          # map(3)
  63        # text(3)
    6C656E  # "len"
  0B        # unsigned(11)
  66        # text(6)
    6F66666736574  # "offset"
  00        # unsigned(0)
  64        # text(4)
    64617461  # "data"
  6B        # text(11)
    68656C6C6F20776F726C64  # "hello world"
```


Links:

```
http://cbor.me/?diag={%20%22len%22:%2011,%20%22offset%22:%200,%20%22data%22:%20%22hello%20world%22%20}
```

# Commands

Each Command ID requires some fields in the packet and responds with some other. Here's the full list of all groups, commands and their parameters that are supported by Zephyr or Mynewt.

Group ID	Command ID	Operation	Required fields	Fields in response
OS (0)  Links:  <a href="#">os_mgmt.h</a>	ECHO (0)	WRITE	<b>d</b> - String  <b>Example:</b>  { "d": "Hello!" }	<b>r</b> - String  <b>Example:</b>  { "r": "Hello!" }
	CONSOLE ECHO CTRL (1)			
	TASKSTAT (2)  <i>Not implemented in Zephyr</i>	READ	N/A	<b>tasks</b> - map of: <ul style="list-style-type: none"><li><b>prio</b> - uint</li><li><b>tid</b> - uint</li><li><b>state</b> - uint</li><li><b>stkuse</b> - uint</li><li><b>stksiz</b> - uint</li><li><b>cswcnt</b> - uint</li><li><b>runtime</b> - uint</li><li><b>last_checkin</b> - uint</li><li><b>next_checkin</b> - uint</li></ul>
	MPSTAT (3)			
	DATETIME STR (4)			
	RESET (5)	WRITE	N/A	N/A
IMAGE (1)  Links:  <a href="#">img_mgmt.h</a>	STATE (0)	READ	N/A	<b>images</b> - map of: <ul style="list-style-type: none"><li><b>image</b> - uint (image number (core id), mainly for nRF53, otherwise 0 or absent). Since NCS 1.7.</li><li><b>slot</b> - uint (0 (primary) or 1 (secondary))</li><li><b>version</b> - String (any, unknown length limit)</li><li><b>hash</b> - byte array (SHA-256 of the image)</li><li><b>bootable</b> - boolean</li><li><b>pending</b> - boolean (true if test or confirm cmd has been sent to that image)</li><li><b>confirmed</b> - boolean (true if the image has booted (confirmed) successfully)</li><li><b>active</b> - boolean (true if currently active, running)</li><li><b>permanent</b> - boolean (true if confirm cmd has been sent to that image)</li></ul> <b>splitStatus</b> - uint (only used in Mynewt)


		WRITE	<b>hash</b> - byte array (optional if confirm = true to confirm the currently running image) <b>confirm</b> - boolean	<i>Same as for READ</i>
	UPLOAD (1)	WRITE	<b>data</b> - byte array <b>len</b> - uint (only when off = 0) <b>off</b> - uint <b>image</b> - image number (core id), Since NCS 1.7. <b>sha</b> - byte array (3 first bytes of 32 byte long SHA-256) ( <a href="#">impl in Java</a> )	<b>rc</b> - uint <b>off</b> - uint
	FILE (2)			
	CORELIST (3)			
	CORELOAD (4)			
	ERASE (5)	WRITE	<b>image</b> - image number (core id), Since NCS 1.7.	<b>rc</b> - uint
	ERASE STATE (6)			
	<b>Links:</b> <a href="#">Android library</a> - I couldn't find it anywhere else <a href="#">iOS Library</a> - same here			
STAT (2)  <b>Links:</b> <a href="#">stat_mgmt.h</a>	SHOW (0)	READ	<b>name</b> - String	<b>rc</b> - uint <b>name</b> - String <b>fields</b> - map of: <ul style="list-style-type: none"> <li>• [entry name] - uint</li> </ul>
	LIST (1)	READ	N/A	<b>stat_list</b> - array of Strings
CONFIG (3)				
LOG (4)  <b>Links:</b> <a href="#">log_mgmt.h</a>  <i>Not implemented in Zephyr</i>	SHOW (0)	READ	<b>log_name</b> - String <b>ts</b> - timestamp (int) <b>index</b> - uint	<b>next_index</b> - uint <b>logs</b> - array of: <ul style="list-style-type: none"> <li>• <b>name</b> - String</li> <li>• <b>type</b> - uint</li> <li>• <b>entries</b> - array of: <ul style="list-style-type: none"> <li>• <b>msg</b> - String</li> <li>• <b>ts</b> - timestamp (int)</li> <li>• <b>level</b> - uint</li> <li>• <b>index</b> - uint</li> <li>• <b>module</b> - uint</li> </ul> </li> </ul> <b>rc</b> - uint
	CLEAR (1)	WRITE	<b>log_name</b> - String	<b>rc</b> - uint 
	APPEND (2)			
	MODULE LIST (3)	READ	N/A	<b>rc</b> - uint <b>module_map</b> - map of: <ul style="list-style-type: none"> <li>• [module name] - uint</li> </ul>
	LEVEL LIST (4)	READ	N/A	<b>rc</b> - uint <b>level_map</b> - map of: <ul style="list-style-type: none"> <li>• [level name] - uint (level)</li> </ul>
	LOGS LIST (5)	READ	N/A	<b>rc</b> - uint <b>log_list</b> - array of Strings (log names)
CRASH (5)				
SPLIT (6)				
RUN (7)				

FS (8)  <b>Links:</b>  <a href="#">fs_mgmt.h</a>	FILE (0)  <b>Links:</b>  <a href="#">fs_mgmt.c</a>	READ	<b>name</b> - String  <b>off</b> - uint	<b>data</b> - byte array  <b>len</b> - uint (only when off = 0)  <b>off</b> - uint  <b>rc</b> - uint
		WRITE	<b>name</b> - String  <b>data</b> - byte array  <b>len</b> - uint (only when off = 0)  <b>off</b> - uint	<b>rc</b> - uint  <b>off</b> - uint
BASIC (63) - Added in NCS 1.7 to erase app settings  <b>Links:</b>  <a href="#">zephyr_groups.h</a>	ERASE_STORAGE (0)  <b>Links:</b>  <a href="#">basic_mgmt.c</a>	WRITE	N/A	<b>rc</b> - uint
PERUSER (64) - reserved for users				

## Errors

An invalid command or an error on the device will be reported with a notification with **rc** (uint) field in the payload.

List of possible return codes:

Name	Return Code	Description
OK	0	OK
UNKNOWN	1	Unknown error
NOMEM	2	No memory
INVAL	3	Invalid value
TIMEOUT	4	Operation timeout
NOENT	5	No entity 
BADSTATE	6	Current state disallows command
MSGSIZE	7	Response too large
NOTSUP	8	Command not supported

**Links:**

<https://github.com/apache/mynewt-mcumgr/blob/master/mgmt/include/mgmt/mgmt.h> mcumgr error codes

## Device Firmware Upgrade

**Update:**

**Multi core update supported since NCS 1.7.** Additional param "image" was added to IMAGE UPLOAD and IMAGE ERASE, where 0 is app core (default) and 1 is net core. The change was mainly to support Thingy:53 and other nRF5340 devices. Each image needs to be sent separately, than the test/command need to be sent for each image, and only than the device needs to be reset. Bin files for different cores are placed in a ZIP file with manifest.json describing which file should be sent with which image parameter.

One of the features of MCU MGR is DFU. A signed image (.bin or .img) file must be created before starting DFU, using the proper certificate. Signing adds the firmware hash and signature to a header before the firmware. Also, a MAGIC value is added that must match to one on the end device.

DFU uses commands from 2 Group IDs: OS (0) and IMAGE (1).

Name	Group ID	Command ID	Operation	Parameters
Validate (list image slots)	IMAGE (1)	STATE (0)	READ	
Upload image	IMAGE (1)	UPLOAD (1)	WRITE	All required
Test (test image with given hash)	IMAGE (1)	STATE (0)	WRITE	<b>confirm</b> = false <b>hash</b>
Confirm (confirm image with given hash)	IMAGE (1)	STATE (0)	WRITE	<b>confirm</b> = true <b>hash</b>
Verify (confirm current slot)	IMAGE (1)	STATE (0)	WRITE	<b>confirm</b> = true (no hash)
Reset	OS (0)	RESET (5)	WRITE	

## Modes

There are 3 modes that an image may be sent:

1. Test and Confirm
2. Test only
3. Confirm only

## Test and Confirm

This mode should be used by default if the firmware supports text/confirm. The client should validate the current slot information, send the new image, send Test command, reset the end device, reconnect to the new image and send Verify command.

## Test only

This mode allows only to test the image. It will not be confirmed and will be reverted to the original one after next reset. The client should validate, send image, send Test command and reset.

## Confirm only

This mode should be used by default if the firmware confirms itself and does not support test. E.g. multi core update in NCS 1.7 does not allow to revert net-core update, so this is the only supported mode for multi-core update. This may also be used if the new image does not support Bluetooth (reconnecting would not be possible), or we are sure it will work. Client should validate, send image, send Confirm command and reset.

## Algorithm

1. Read hash of the firmware to be sent.
2. Ensure the MTU is set to highest possible value.
3. Send *Verify* command to receive current slot information.
  1. Check if slot 0's hash matches with the hash. If so, based on the mode and the image properties it could be *Verified* or DFU is complete.
  2. Check if slot 1's (if there is such) hash matches with the hash. If so, send *Test* or *Confirm* command, based on the mode.
4. Send the new image using *Upload* command. Many commands will have to be sent. As a response to each one the end device will confirm the offset.
5. Send *Test* or *Confirm* command, based on the mode.
6. Send *Reset* command.
7. If **Test and Confirm** mode was used, reconnect to the device. Otherwise upload is complete.
8. Send *Verify* command. This will confirm the image in slot 0. If the test succeeded, and images were successfully swapped on reset, this will confirm the tested image. Otherwise it will confirm already confirmed image and the invalid one (e.g. signed with a wrong certificate) has been removed and slot 1 is empty.
9. Based on the result - it's either success, or some failure. Device may be test again to apply changes, but it's already running the proper image.

## Mobile clients

iOS MCUMGR Library and Sample app may be found here: <https://github.com/JuulLabs-OSS/mcumgr-ios> <https://github.com/NordicSemiconductor/IOS-nRF-Connect-Device-Manager>

Android MCUMGR Library and Sample app may be found here: <https://github.com/JuulLabs-OSS/mcumgr-android> <https://github.com/NordicSemiconductor/Android-nRF-Connect-Device-Manager>

