WP-RM-117 – Wirepas Mesh Remote API Reference Manual

Reference Manual

Version: v5.0A

Applies to Wirepas Mesh firmware release v5.0 onwards

This document describes the Remote API provided by the Wirepas Mesh stack.

Confidential

Table of Contents

1. Introduction

Wirepas Mesh stack v3.1 onwards provides a way to configure and reconfigure nodes remotely, over the Wirepas Mesh network. Parameters such as node address, network address and node role can be set or queried. This feature is called the Remote API.

The Remote API refers to node parameters in the same way the Wirepas Mesh Dual-MCU API does. Parameters are called attributes and are separated into MSAP (Management Services) and CSAP (Configuration Services). Familiarity with document *WP-RM-100 – Wirepas Mesh Dual-MCU API Reference Manual* [1] is required to understand the concepts in this document.

2. Operation

The Wirepas Mesh Remote API requests and responses are sent as regular data packets in the Wirepas Mesh network. One or more requests can be combined into a Remote API request packet.

Using the sink node, a request packet is sent to target nodes and the target nodes then send a response packet back to the sink. Only sinks can send Remote API requests via the dual-MCU API. A single-MCU application must make sure it only sends Remote API requests from a sink node.

2.1. Request Packets and Responses

Requests are sent from a sink to a target node as regular data packets. Multiple nodes can be addressed at the same time by using the broadcast address as the destination (however, see section 3.9 for an exception). The Wirepas Mesh stack uses certain reserved source and destination endpoints for identifying Remote API request packets, see Table 1.

Table 1: Reserved Endpoints for Wirepas Mesh Remote API

	Source Endpoint	**Destination Endpoint**
Request Packet	255 (0xFF in hexadecimal)	240 (0xF0 in hexadecimal)
Response Packet	240 (0xF0 in hexadecimal)	255 (0xFF in hexadecimal)

When a node receives a request packet that has the correct source and destination endpoints, it processes the requests in the packet and forms a response packet. The response packet is sent to the AnySink address. Response packets have the same reserved endpoints that the request packets have, except in reverse.

Remote API request packets can be sent with either normal or high priority Quality of Service (QoS) class. The response packet uses the same QoS class as the request packet.

2.2. Packet Structure

Each Remote API request packet may contain one or more requests. The requests use a Type-Length-Value format, outlined in Figure 1.

Octet 0	Octet 1	Octets 2 .. n
Type	Length	Value

Figure 1: Structure of a Request

The Type octet identifies the request. It is followed by a Length octet, which indicates how many octets the value occupies. If the Length is non-zero, a number of Value octets follow. Not all requests have values associated with them. In that case, the Length octet is zero and the complete request is only two octets long, i.e. just the Type and Length octets.

Multiple requests can simply be concatenated together, up to the maximum size of a data packet. The maximum size of a data packet depends on firmware version and enabled features. It can be queried by reading CSAP attribute 5: cMTU. See [1] for details.

When a node processes a request packet, it constructs a response packet containing responses for each processed request in the request packet. Responses use the same Type-Length-Value format as requests.

In case of an error, processing stops at the first invalid request. An error response is added to the response packet and rest of the requests in the request packet are ignored.

The type octet is also used to differentiate a request from a response, see Table 2.

Table 2: Request and Response Types

Type (Decimal)	Type (Hexadecimal)	Purpose
0 .. 123	0x00 .. 0x7B	Request
124 .. 127	0x7C .. 0x7F	Reserved for future use
128 .. 251	0x80 .. 0xFB	Response
252 .. 255	0xFC .. 0xFF	Error response

Values may occupy more than one octet. Any such values are stored in little endian order, i.e. the least significant octet first.

2.3. Writing MSAP and CSAP Attributes

An overview of the states of configuring one or more MSAP and CSAP attributes is shown in Figure 2.

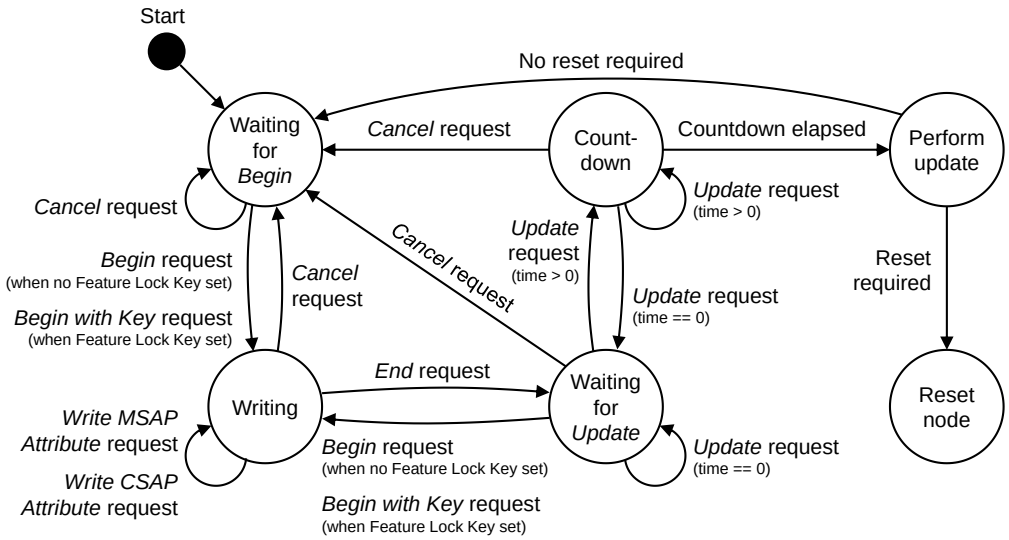

Figure 2: States of Configuring MSAP and CSAP Attributes

Requests to write to MSAP and CSAP attributes (sections 3.7 and 3.9) do not modify the values of the attributes directly. Instead, the values are written in a temporary buffer which can then be taken in to use with an *Update* request (see section 3.6) later. Alternatively, a *Cancel* request (see section 3.5) can be used to discard any buffered values.

An *Update* request is typically sent afterwards in its own packet. The *Update* request has a countdown timer, so that the update can be triggered with a delay. This is to make sure that, on large networks, even the most remote nodes will have received the *Update* request before nodes in between have changed their parameters.

Writing attributes happens between *Begin* and *End* requests (sections 3.2, 3.3 and 3.4). It is not mandatory to have the *Begin* and *End* appear on the same request packet, but it is recommended. Otherwise, packets that appear out-of-order may confuse the Remote API request processing.

In order to support keeping features securely locked, the Remote API handles feature lock key access differently than the dual-MCU API. See [1] for description of CSAP attributes cFeatureLockBits and cFeatureLockKey.

A separate *Begin with Lock* request (see section 3.3) allows writing to MSAP and CSAP attributes without having to explicitly unlock the feature lock first. Between the *Begin with Lock* request and *End* request, requests to write MSAP and CSAP attributes succeed (provided that the key was correct in the *Begin with Lock* request), even if feature lock bits would otherwise prevent writing of MSAP or CSAP attributes. *Begin with Lock* request only affects Remote API. It does not let the Dual-MCU API write attributes (if the feature lock bits and key are set to prevent it), for example.

More so than with normal *Begin* request, it is suggested that the *Begin with Lock*, *Write* and *End* requests are all packed in one request packet, so that the feature lock bits do not remain open across request packets. This prevents any packets without a key from modifying MSAP and CSAP attributes.

2.4. Reading MSAP and CSAP Attributes

Reading MSAP and CSAP attributes can happen at any time. The read requests (sections 3.8 and 3.10) do not have to be between *Begin* and *End* requests. This implies, that reading MSAP or CSAP attributes is not possible if the feature lock bits and key have been set to prevent reading those attributes.

3. Requests and Responses

Remote API requests are listed in Table 3 and explained in detail below.

Table 3: Request and Response Types

Request (Hexadecimal)	Response (Hexadecimal)	Description
0x00	0x80	Ping
0x01	0x81	Begin
0x02	0x82	Begin with Lock
0x03	0x83	End
0x04	0x84	Cancel
0x05	0x85	Update
0x0B	0x8B	Write MSAP Attribute
0x0C	0x8C	Read MSAP Attribute
0x0D	0x8D	Write CSAP Attribute
0x0E	0x8E	Read CSAP Attribute
0x19	0x99	MSAP Scratchpad Status
0x1A	0x9A	MSAP Scratchpad Update
0x4F	0xCF	MSAP Max Queue Time Write
0x50	0xD0	MSAP Max Queue Time Read
0x61	0xE1	MSAP Enable Joining Beacon
0x62	0xE2	MSAP Disable Joining Beacon
0x63	0xE3	MSAP Joining Beacon Status

In case there is an error carrying out a request, an error response is added to the response packet and no further requests from the request packet are processed.

Error responses are listed in Table 4 and explained in detail below.

Table 4: Error Responses

Response (Hexadecimal)	Description
0xF8	Access Denied
0xF9	Write-only Attribute
0xFA	Invalid Broadcast Request
0xFB	Invalid Begin
0xFC	No Space for Response
0xFD	Invalid Value
0xFE	Invalid Length
0xFF	Unknown Request

3.1. Ping

The Ping request does nothing, except causes a response to be returned. This can be used to determine if a node is running Wirepas Mesh stack that supports Remote API. Remote API was introduced in firmware version 3.1.

Starting from firmware version 3.4, there can be an optional payload of up to 16 octets, which is returned unmodified in the response. By using a unique identifier as the payload, responses can be matched with requests, e.g. when sending multiple requests in rapid succession. The Ping request can also be used to do simple latency testing, e.g. by placing a timestamp in the payload and comparing it to the time a corresponding Ping response is received.

The Ping request can be issued at any time. Format of the Ping request is shown in Figure 3 and the response in Figure 4.

Octet 0	Octet 1	Octets 2 .. 17
Type: 0x00	Length	Payload

Figure 3: Ping Request

Octet 0	Octet 1	Octets 2 .. 17
Type: 0x80	Length	Payload

Figure 4: Ping Response

3.2. Begin

Configuring MSAP and CSAP attributes is only possible between a Begin request (or Begin with Lock request, section 3.3) and End request (section 3.4). The Begin request allocates a temporary buffer for storing values, which can then be written using the Write MSAP Attribute (section 3.7) and Write CSAP Attribute (section 3.9) requests.

Format of the Begin request is shown in Figure 5 and the response in Figure 6.

Octet 0	Octet 1
Type: 0x01	Length: 0x00

Figure 5: Begin Request

Octet 0	Octet 1
Type: 0x81	Length: 0x00

Figure 6: Begin Response

Whether to use a Begin or Begin with Lock request depends on the feature lock key. If a key is set, only a Begin with Lock request is accepted. If no key is set, only the Begin request can be used. See the description of CSAP attribute cFeatureLockKey in [1].

If a feature lock key is set, the Begin request is rejected with an Access Denied error response (section 0). If an update countdown is running, the request is rejected with an Invalid Begin error response (section 3.21). If there is not enough memory for a temporary buffer, an Invalid Value error response is returned (section 3.23).

It is possible to issue multiple Begin requests before an End request. Any extraneous requests before the first are simply ignored.

3.3. Begin with Lock

The Begin with Lock request is identical to the Begin request (section 3.2), except that it allows temporarily bypassing the feature lock. If a feature lock key is set, only the Begin with Key request is accepted and the Begin request is not. Likewise, only the Begin request can be used if no feature lock key is set. See the description of CSAP attribute cFeatureLockKey in [1].

Format of the Begin with Key request is shown in Figure 7 and the response in Figure 8.

Octet 0	Octet 1	Octets 2 .. 17
Type: 0x02	Length: 0x10	Feature Lock Key

Figure 7: Begin with Lock Request

Octet 0	Octet 1
Type: 0x82	Length: 0x00

Figure 8: Begin with Lock Response

The Begin with Key request only affects the Write MSAP Attribute (section 3.7) and Write CSAP Attribute (section 3.9) requests, and only until the next End request (section 3.4). Furthermore, Begin with Lock only affects the Remote API, and does not affect the lock state of Dual-MCU API, for example. For longer-term unlocking of the feature lock, use the Write CSAP Request to set the CSAP attribute cFeatureLockKey to FF FF .. FF (hexadecimal), i.e. all bits set. See section 4 for an example.

If the given key is incorrect, or if no feature lock key is set, the Begin with Key request is rejected with an Access Denied error (section 0). If an update countdown is running, the request is rejected with an Invalid Begin error (section 3.21). If there is not enough memory for a temporary buffer, an Invalid Value error response is returned (section 3.23).

It is possible to issue multiple Begin with Key requests before an End request. Any extraneous requests before the first are simply ignored.

3.4. End

After configuring MSAP or CSAP attributes, and End request must be issued. Format of the End request is shown in Figure 9 and the response in Figure 10.

Octet 0	Octet 1
Type: 0x03	Length: 0x00

Figure 9: End Request

Octet 0	Octet 1
Type: 0x83	Length: 0x00

Figure 10: End Response

If no valid Begin request has been issued, an Invalid Begin error response is returned (section 3.21).

It is possible to issue multiple Begin / End request sets before sending an Update request (section 3.6). This way, configuring of attributes can be distributed across many request packets without having to leave the feature lock temporarily open (see the description of Begin with Lock request, section 3.3).

3.5. Cancel

A Cancel request discards all CSAP and MSAP attribute values stored in a temporary buffer and cancels a running update countdown. Cancel requests can be issued at any time. It never returns an error response.

Format of the Cancel request is shown in Figure 11 and the response in Figure 12.

Octet 0	Octet 1
Type: 0x04	Length: 0x00

Figure 11: Cancel Request

Octet 0	Octet 1
Type: 0x84	Length: 0x00

Figure 12: Cancel Response

3.6. Update

An Update request starts a countdown, after which the MSAP and CSAP attributes are written from a temporary buffer to the actual attributes. The node is then reset, if any of the written attributes require a reboot. See Table 6 and Table 7 for a list of attributes that require a reboot.

Format of the Update request is shown in Figure 13 and the response in Figure 14.

Octet 0	Octet 1	Octet 2	Octet 3
Type: 0x05	Length	Time LSB	Time MSB

Figure 13: Update Request

Octet 0	Octet 1	Octet 2	Octet 3
Type: 0x85	Length: 0x02	Time LSB	Time MSB

Figure 14: Update Response

The countdown time value range is described in Table 5. A special value of 0 is also accepted. It cancels a running countdown. If the countdown time is omitted completely, the current countdown time is reported, or zero if no countdown is running.

Due to the way nodes scan for neighbors right after reboot, it is better for the nodes not to reboot at the same time. To facilitate that, a random delay between 0 and 20 seconds is added to the countdown value. The resulting countdown value is placed in the response.

Table 5: Countdown Time Values

Minimum Value	Maximum Value
10 (10 seconds)	32767 (9 hours, 6 minutes)

If no Begin and End requests were issued, an Invalid Begin error response is returned (section 3.21). If the countdown time is not valid, an Invalid Value error response is returned (section 3.23). If the feature lock key was changed (e.g. via Dual-MCU API) after the Begin with Lock request (section 3.3) was issued, an Access Denied error response is returned (section 0).

3.7. Write MSAP Attribute

The Write MSAP Attribute request allows configuring MSAP attributes. The supported attributes are listed in Table 6. Currently only one MSAP attribute is supported.

When an Update request (section 3.6) is issued and after the countdown time has elapsed, the node is rebooted if any of the attributes that require a reboot is being configured. The sole supported MSAP attribute does not require a reboot.

The Write MSAP Attribute is allowed only after a Begin (section 3.2) or Begin with Lock request (section 3.3) and before an End request (section 3.4). Format of the Write MSAP Attribute request is shown in Figure 15 and the response in Figure 16.

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x0B	Length	Attr ID LSB	Attr ID MSB	Value

Figure 15: Write MSAP Attribute Request

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x8B	Length	Attr ID LSB	Attr ID MSB	Value

Figure 16: Write MSAP Attribute Response

Table 6: Supported MSAP Attributes

Attribute ID	Attribute Name	Value Length (Octets)	Reboot Required?
9	mAccessCycleRange	4	No

If no valid Begin request has been issued, an Invalid Begin error response is returned (section 3.21). If the number of octets in the value is incorrect, an Invalid Length error response is returned (section 3.24). In case of a faulty value, an Invalid Value error response is returned (section 3.23). If the attribute ID is not recognized, an Unknown Request error response is returned (section 3.25).

3.8. Read MSAP Attribute

Reading MSAP attributes is done using the Read MSAP Attribute request. The same attribute is supported as with the Write MSAP Attribute request, listed in Table 6.

The Read MSAP Attribute request can be issued at any time. It does not need to be between Begin and End requests. The value returned is the current value of the attribute. There is no way to read the value in the temporary buffer, which is set using the Write MSAP Attribute request (section 3.7).

Format of the Read MSAP Attribute request is shown in Figure 17 and the response in Figure 18.

Octet 0	Octet 1	Octet 2	Octet 3
Type: 0x0C	Length: 0x02	Attr ID LSB	Attr ID MSB

Figure 17: Read MSAP Attribute Request

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x8C	Length	Attr ID LSB	Attr ID MSB	Value

Figure 18: Read MSAP Attribute Response

If the attribute ID is not recognized, an Unknown Request error response is returned (section 3.25). If feature lock bits prevent reading MSAP attributes, an Access Denied error response is returned (section 0). If a value is not set for an attribute, an Invalid Value error response is returned (section 3.23).

3.9. Write CSAP Attribute

The Write CSAP Attribute request is used for configuring CSAP attributes. The supported attributes are listed in Table 7.

When an Update request (section 3.6) is issued and after the countdown time has elapsed, the node is rebooted if any of the attributes that require a reboot is being configured. Configuring most CSAP attributes require a reboot.

The Write CSAP Attribute is allowed only after a Begin (section 3.2) or Begin with Lock request (section 3.3) and before an End request (section 3.4). Format of the Write CSAP Attribute request is shown in Figure 19 and the response in Figure 20.

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x0D	Length	Attr ID LSB	Attr ID MSB	Value

Figure 19: Write CSAP Attribute Request

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x8D	Length	Attr ID LSB	Attr ID MSB	Value

Figure 20: Write CSAP Attribute Response

Table 7: Supported CSAP Attributes

Attribute ID	Attribute Name	Value Length (Octets)	Reboot Required?
1	cNodeAddress (see Note 1)	4	Yes
2	cNetworkAddress	3	Yes
3	cNetworkChannel	1	Yes
4	cNodeRole (see Note 2)	1 or 2 (see Note 4)	Yes
13	cCipherKey (see Note 3)	16	Yes
14	cAuthenticationKey (see Note 3)	16	Yes
20	cOfflineScan	2	No
21	cChannelAllocMap	4	Yes
22	cFeatureLockBits	4 or 8 (see Note 4)	No
23	cFeatureLockKey (see Note 3)	16	No

Note 1: To prevent accidental misconfiguration of node addresses, the cNodeAddress attribute can only be set using a unicast request packet. Issuing a Write CSAP Attribute request for cNodeAddress in a broadcast packet will result in an Invalid Broadcast Request error response (section 3.20).

Note 2: To prevent accidental misconfiguration of node role, setting a sink to a headnode or subnode, and vice versa, is not permitted. An Invalid Value error response is returned in such a case (section 3.23).

Note 3: Responses for valid Write CSAP Attribute requests normally contain both the attribute ID and the new value. However, attributes with keys (cCipherKey, cAuthenticationKey and cFeatureLockKey) only contain the attribute ID in the response. This is to reduce the likelihood of accidentally disclosing the keys.

Note 4: Starting from firmware version 3.4, the cNodeRole and cFeatureLockBits attributes can have an optional bit mask after the value. Any zero bits in the mask will keep the bits from the original value. Only bits set in the mask will be affected. See section 4 for an example.

If no valid Begin request has been issued, an Invalid Begin error response is returned (section 3.21). If the number of octets in the value is incorrect, an Invalid Length error response is returned (section 3.24). In case of a faulty value, an Invalid Value error response is returned (section 3.23). If the attribute ID is not recognized, an Unknown Request error response is returned (section 3.25).

3.10. Read CSAP Attribute

Reading CSAP attributes is done using the Read CSAP Attribute request. Supported attributes are the same as with the Write CSAP Attribute request, listed in Table 7.

The Read CSAP Attribute request can be issued at any time. It does not need to be between Begin and End requests. The value returned is the current value of the attribute. There is no way to read the value in the temporary buffer, which is set using the Write CSAP Attribute request (section 3.9).

Format of the Read CSAP Attribute request is shown in Figure 21 and the response in Figure 22.

Octet 0	Octet 1	Octet 2	Octet 3
Type: 0x0E	Length: 0x02	Attr ID LSB	Attr ID MSB

Figure 21: Read CSAP Attribute Request

Octet 0	Octet 1	Octet 2	Octet 3	Octets 4 .. n
Type: 0x8E	Length	Attr ID LSB	Attr ID MSB	Value

Figure 22: Read CSAP Attribute Response

If the attribute ID is not recognized, an Unknown Request error response is returned (section 3.25). If feature lock bits prevent reading CSAP attributes, an Access Denied error response is returned (section 0). If a value is not set for an attribute, an Invalid Value error response is returned (section 3.23).

Reading keys (cCipherKey, cAuthenticationKey and cFeatureLockKey) is not possible. If a key is set, a Write-only Attribute error response is returned. If no key is set, the Invalid Value error response is returned, like for any other CSAP attribute that has no value.

3.11. MSAP Scratchpad Status

As an alternative to the out-of-band MSAP Remote Status and MSAP Remote Update requests (see [1] for details), Remote API supports similar functionality from firmware version 3.4 onwards. The old MSAP Remote Status and MSAP Remote Update requests were removed in firmware version 5.0.

The MSAP Scratchpad Status request can be issued at any time. It does not need to be between Begin and End requests. The response is similar but not identical to the old MSAP Remote Status Indication (see [1] for details). The response has either 24 or 39 octets, depending on whether the firmware supports reporting the application area information. The 39-octet response was introduced in firmware version v4.0.

Format of the MSAP Scratchpad Status request is shown in Figure 23 and the response in Figure 24. Fields of the response are explained in Table 8 and

Table 9.

Octet 0	Octet 1
Type: 0x19	Length: 0x00

Figure 23: MSAP Scratchpad Status Request

Octet 0	Octet 1	Octets 2 .. 5		
Type: 0x99	Length	Stored Scratchpad Number of Bytes		
Octets 6 .. 7		Octet 8	Octet 9	Octet 10
Stored Scratchpad CRC		St Scr Seq	St Scr Type	St Scr Status
Octets 11 .. 14				Octets 15 .. 16
Processed Scratchpad Number of Bytes				Processed Scratchpad CRC
Octet 17	Octets 18 .. 21			
Proc Scr Seq	Processed Firmware Area ID			
Octet 22	Octet 23	Octet 24	Octet 25	
FW Major Ver	FW Minor Ver	FW Maint Ver	FW Devel Ver	
Octets 26 .. 29				Octets 30 .. 31
Application Processed Scratchpad Number of Bytes				App Proc Scratchpad CRC
Octet 32	Octets 33 .. 36			
App P Scr Seq	Processed Application Area ID			
Octet 37	Octet 38	Octet 39	Octet 40	
App Major Ver	App Minor Ver	App Maint Ver	App Devel Ver	

Figure 24: MSAP Scratchpad Status Response

Table 8: MSAP Scratchpad Status Response Fields

Start Octet	Value Length (Octets)	Description
0	1	Response type: 0x99
1	1	Length in octets, either 24 (0x18) or 39 (0x27)
2	4	Length of the scratchpad present in the node, in octets, or 0 if no scratchpad is present in the node
6	2	CRC of the scratchpad present in the node, or 0 if there is no scratchpad present in the node
8	1	Sequence number of the scratchpad present in the node, or 0 if there is no scratchpad present in the node
9	1	Type of the scratchpad present in the node: • 0 = Blank: No valid scratchpad is present • 1 = Present: A valid scratchpad is present, but has not been marked to be processed • 2 = Process: A valid scratchpad is present and has been marked to be processed
10	1	Status of the scratchpad present in the node: • 255 = New: Bootloader has not yet processed the scratchpad • 0 = Success: Bootloader has processed the scratchpad successfully • 1 .. 254 = Error: Bootloader encountered an error while processing the scratchpad Status is 0 also if there is no scratchpad present in the node
11	4	Length, in octets, of the scratchpad that produced the firmware currently running on the node
15	2	CRC of the scratchpad that produced the firmware currently running on the node
17	1	Sequence number of the scratchpad that produced the firmware currently running on the node
18	4	Memory area ID of the file in the scratchpad that produced the firmware currently running on the node. Scratchpads may contain multiple firmware images. This value can be used to determine which one the bootloader picked.
22	1	Major version number of currently running firmware
23	1	Major version number of currently running firmware
24	1	Maintenance version number of currently running firmware
25	1	Development version number of currently running firmware

Table 9: Extended MSAP Scratchpad Status Response Fields

Start Octet	Value Length (Octets)	Description
26	4	Length, in octets, of the scratchpad that produced the application currently running on the node
30	2	CRC of the scratchpad that produced the application currently running on the node
32	1	Sequence number of the application that produced the firmware currently running on the node
33	4	Memory area ID of the file in the scratchpad that produced the application currently running on the node. Scratchpads may contain multiple application images. This value can be used to determine which one the bootloader picked.
37	1	Major version number of currently running application
38	1	Major version number of currently running application
39	1	Maintenance version number of currently running application
40	1	Development version number of currently running application

This request does not return error responses, except when there is no space for the fairly large response, in which case a No Space for Response error response is returned (section 3.22).

3.12. MSAP Scratchpad Update

The MSAP Scratchpad Update request marks the scratchpad for processing by the bootloader. After an Update request (section 3.6) is issued and when the countdown reaches zero, the scratchpad is marked for processing and the node is rebooted. This causes the bootloader to go through the scratchpad contents and update any matching stack firmware and application in the node.

The MSAP Scratchpad Update request is allowed only after a Begin (section 3.2) or Begin with Lock request (section 3.3) and before an End request (section 3.4). Format of the MSAP Scratchpad Update request is shown in Figure 25 and the response in Figure 26.

Octet 0	Octet 1	Octet 2
Type: 0x1A	Length: 0x01	Scr Sequence

Figure 25: MSAP Scratchpad Update Request

Octet 0	Octet 1	Octet 2
Type: 0x9A	Length: 0x01	Scr Sequence

Figure 26: MSAP Scratchpad Update Response

The sole parameter is the scratchpad sequence number. If the stored scratchpad has a different sequence number, or if there is no stored scratchpad, an Invalid Value error response is returned (section 3.23). If no valid Begin request has been issued, an Invalid Begin error response is returned (section 3.21).

3.13. MSAP Max Queue Time Write

The MSAP Max Queue Time Write request is used to change the current value for how long a message is hold in the node's queue before it is discarded. Queuing time can be changed to normal and high priority messages separately.

Select queuing time carefully, too low a value may cause messages to be discarded unnecessarily and too high a value will result in filling up the message queues. For consistent performance it is recommended to use the same queuing times for the whole network.

Minimum queuing time shall be bigger than access cycle interval in TDMA networks. It is recommended to use multiples of access cycle interval (+ extra) to give time for message repetitions, higher priority messages taking over the access slot etc. Access cycle is not limiting the minimum value in CSMA-CA networks.

Precision of the time when message is discarded depends on the checking interval of message's age. Interval is 1s in CSMA-CA and 15s for energy saving reasons in TDMA networks i.e. precision is 1s or 15s depending on used channel access method.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0x4F	Length: 0x03	Priority	Time LSB	Time MSB

Figure 27: MSAP Max Queue Time Write Request

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xCF	Length: 0x03	Priority	Time LSB	Time MSB

Figure 28: MSAP Max Queue Time Write Response

3.14. MSAP Max Queue Time Read

The MSAP Max Queue Time Read request is used to read the current value of how long a message is hold in the node's queue before it is discarded. Queuing time for normal and high priority messages have their own values.

Octet 0	Octet 1	Octet 2
Type: 0x50	Length: 0x01	Priority

Figure 29: MSAP Max Queue Read Request

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xD0	Length: 0x03	Priority	Time LSB	Time MSB

Figure 30: MSAP MAX Queue Read Response

3.15. MSAP Enable Joining Beacon

The MSAP Enable Joining Beacon request is used to configure and enable periodic transmission of joining beacons. Joining beacons are part of a feature called Open Joining, which was introduced in Wirepas Mesh firmware version 5.0. See [2] for details.

The MSAP Enable Joining Beacon request can be issued at any time. It does not need to be between Begin and End requests. Format of the MSAP Enable Joining Beacon request is shown in Figure 31 and the response in Figure 32. Fields of the request are explained in Table 10.

Octet 0	Octet 1	Octets 2 .. 5		
Type: 0x61	Length	Beacon Interval in µs, LSB First		
Octets 6 .. 9			Octet 10	Octet 11
Radio Address, LSB First			Channel	Power
Octets 12 .. 15			Octets 16 .. n	
Joining Type, LSB First			Payload	

Figure 31: MSAP Enable Joining Beacon Request

Octet 0	Octet 1	Octets 2 .. 5		
Type: 0xE1	Length	Beacon Interval in µs, LSB First		
Octets 6 .. 9			Octet 10	Octet 11
Radio Address, LSB First			Channel	Power
Octets 12 .. 15			Octets 16 .. n	
Joining Type, LSB First			Payload	

Figure 32: MSAP Enable Joining Beacon Response

Table 10: MSAP Enable Joining Beacon Request and Response Fields

Start Octet	Value Length (Octets)	Description
0	1	Type, request: 0x61, response: 0xE1
1	1	Length in octets, 14 (0x0E) to 78 (0x4E)
2	4	Beacon interval in microseconds
6	4	Radio address (network address) to use for sending joining beacons
10	1	Channel to use for sending joining beacons
11	1	Power in dBm to use for sending joining beacons, a signed 8-bit value
12	4	Joining type, a 32-bit value that identifies the supported joining protocol
8	0 to 64	Optional payload

3.16. MSAP Disable Joining Beacon

The MSAP Disable Joining Beacon request stops the transmission of periodic joining beacons that were enabled with MSAP Enable Joining Beacon request, above. The request takes no parameters. Format of the MSAP Disable Joining Beacon request is shown in Figure 33 and the response in Figure 34.

Octet 0	Octet 1
Type: 0x62	Length: 0x00

Figure 33: MSAP Disable Joining Beacon Request

Octet 0	Octet 1
Type: 0xE2	Length: 0x00

Figure 34: MSAP Disable Joining Beacon Response

3.17. MSAP Joining Beacon Status

Querying the status of the joining beacon transmissions is done using the MSAP Joining Beacon Status request. The request takes no parameters. A single status octet is returned in the response: 1 if joining beacon transmissions is enabled, 0 if not.

Format of the MSAP Joining Beacon Status request is shown in Figure 35 and the response in Figure 36.

Octet 0	Octet 1
Type: 0x63	Length: 0x00

Figure 35: MSAP Joining Beacon Status Request

Octet 0	Octet 1	Octet 2
Type: 0xE3	Length: 0x01	Status

Figure 36: MSAP Joining Beacon Status Response

3.18. Error: Access Denied

Access Denied error response is returned in the following situations:

- For Read MSAP Attribute request, if feature lock bits prevent reading MSAP attributes
- For Read CSAP Attribute request, if feature lock bits prevent reading CSAP attributes
- For Begin request, if a feature lock key is set
- For Begin with Lock request, if a feature lock key is not set, or the key given is incorrect
- For Update request, if the feature lock key has changed after the last Begin with Lock request (e.g. via the Dual-MCU API)
- For MSAP Joining Beacon requests (Enable, Disable, Status), if the Remote API control of the joining beacons is disabled by the application

See [1] for description of CSAP attributes cFeatureLockBits and cFeatureLockKey.

Format of the Access Denied error response is shown in Figure 37. It contains the request type that failed and optionally the ID of the attribute that could not be accessed.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xF8	Length	Request	Attr ID LSB	Attr ID MSB

Figure 37: Access Denied Response

3.19. Error: Write-only Attribute

Write-only Attribute error response is returned if a key attribute (cCipherKey, cAuthenticationKey and cFeatureLockKey) is read with the Read CSAP Attribute request (section 3.10), and the key is set. If the key is not set, an Invalid Value error response is returned instead (section 3.23).

Format of the Write-only Attribute error response is shown in Figure 38. It contains the request type that failed (i.e. Read CSAP Attribute, section 3.10) and the ID of the key attribute that could not be read.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xF9	Length: 0x03	Request: 0x0E	Attr ID LSB	Attr ID MSB

Figure 38: Write-only Attribute Response

3.20. Error: Invalid Broadcast Request

Setting node address is not possible using a broadcast request packet. An Invalid Broadcast Request error response is returned if a Write CSAP Attribute request with cNodeAddress attribute ID is issued in a broadcast request packet.

Format of the Invalid Broadcast Request error response is shown in Figure 39. It contains the request type that failed (i.e. Write CSAP Attribute, section 3.9) and the ID of the key attribute that could not be written (i.e. 1, cNodeAddress).

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFA	Length: 0x03	Request: 0x0D	Attr ID LSB	Attr ID MSB

Figure 39: Invalid Broadcast Request Response

3.21. Error: Invalid Begin

Invalid Begin error response is returned in the following situations:

- For Begin or Begin with Lock request, if an update countdown is running
- For Write MSAP Attribute or Write CSAP Attribute if no Begin request has been issued
- For End request, if no Begin or Begin with Lock request has been issued
- For Update request, if no Begin and End requests have been issued

Format of the Invalid Begin error response is shown in Figure 40. It contains the request type that failed and optionally the ID of the attribute that could not be written.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFB	Length	Request	Attr ID LSB	Attr ID MSB

Figure 40: Invalid Begin Response

3.22. Error: No Space for Response

Responses to requests in one request packet are always sent in a single response packet. If there is not enough space for a response in the response packet, the request is not processed and the No Space for Response error response is returned instead. No requests after the failing request are processed. If there are less than five octets of space left in the response packet, the No Space for Response error response is omitted.

Format of the No Space for Response error response is shown in Figure 40. It contains the request type that failed and optionally the ID of the attribute that could not be read or written.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFC	Length	Request	Attr ID LSB	Attr ID MSB

Figure 41: No Space for Response, Response

3.23. Error: Invalid Value

An Invalid Value error response is returned in the following situations:

- For Begin or Begin with Lock request, if memory for a temporary buffer could not be allocated
- For Update request, if the countdown time value is invalid

- For Write MSAP Attribute or Write CSAP Attribute, if the attribute value is invalid
- For Read MSAP Attribute or Read CSAP Attribute, if the attribute value is not set
- For MSAP Max Queue Time Read, if priority is invalid
- For MSAP Max Queue Time Write, if priority or time is invalid
- For MSAP Scratchpad Update, if the sequence is incorrect, or there is no scratchpad stored
- For MSAP Enable Joining Beacon, if any of the parameters are incorrect

Format of the Invalid Value error response is shown in Figure 42. It contains the request type and optionally the ID of the attribute in the failed request.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFD	Length	Request	Attr ID LSB	Attr ID MSB

Figure 42: Invalid Value Response

3.24. Error: Invalid Length

If a request has wrong number of octets, an Invalid Length error response is returned. Reasons for the error are:

- The request packet is shorter than what the Length field of the request would indicate.
- The Length field is incorrect for a request.
- Attribute value is of the wrong length.

Format of the Invalid Length error response is shown in Figure 43. It contains the request type and for attribute write and read requests, the attribute ID.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFE	Length	Request	Attr ID LSB	Attr ID MSB

Figure 43: Invalid Length Response

3.25. Error: Unknown Request

Any request that is not recognized will result in an Unknown Request response. Attribute reading and writing with an unrecognized attribute ID will also cause the Unknown Request response to be returned.

Format of the Unknown Request error response is shown in Figure 44. It contains the request type that could not be recognized, or a request type and the ID of the attribute that could not be recognized.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4
Type: 0xFF	Length	Request	Attr ID LSB	Attr ID MSB

Figure 44: Unknown Request Response

4. Example Remote API Request and Response Packets

The example below configures a node with the following settings:

- Node address: 0x123456
- Network address: 0xABCDEF

- Network channel: 10
- Node role: low-latency headnode with automatic role selection (0x92)
- Encryption key: 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0 00
- Authentication key: 00 F0 E0 D0 C0 B0 A0 90 80 70 60 50 40 30 20 10
- Feature lock bits: prevent reading and writing MSAP and CSAP attributes, prevent factory reset (0xFFFFEC3F)
- Feature lock key: FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00

A complete request packet, 99 octets, is as follows (octets shown in hexadecimal):

```
04 00 01 00 0D 06 01 00 56 34 12 00 0D 05 02 00 EF CD AB 0D 03 03 00
0A 0D 03 04 00 92 0D 12 0D 00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0
E0 F0 00 0D 12 0E 00 00 F0 E0 D0 C0 B0 A0 90 80 70 60 50 40 30 20 10
0D 06 16 00 3F EC FF FF 0D 12 17 00 FF EE DD CC BB AA 99 88 77 66 55
44 33 22 11 00 03 00
```

The request packet is deconstructed and explained in Table 11.

Table 11: A Request Packet

Octets (Hexadecimal)	Description
04 00	Issue a Cancel request, to put Remote API in a known state.
01 00	Issue a Begin request.
0D 06 01 00 56 34 12 00	Write 0x123456 to CSAP attribute cNodeAddress
0D 05 02 00 EF CD AB	Write 0xABCDEF to CSAP attribute cNetworkAddress
0D 03 03 00 0A	Write 0x0A to CSAP attribute cNetworkChannel
0D 03 04 00 92	Write 0x92 to CSAP attribute cNodeRole
0D 12 0D 00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0 00	Write a key to CSAP attribute cCipherKey
0D 12 0E 00 00 F0 E0 D0 C0 B0 A0 90 80 70 60 50 40 30 20 10	Write a key to CSAP attribute cAuthenticationKey
0D 06 16 00 3F EC FF FF	Write lock bits to CSAP attribute cFeatureLockBits
0D 12 17 00 FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00	Write a key to CSAP Attribute cFeatureLockKey
03 00	Issue an End request.

If everything goes well, the node will respond with 51 octets (shown here in hexadecimal):

```
84 00 81 00 8D 06 01 00 56 34 12 00 8D 05 02 00 EF CD AB 8D 03 03 00
0A 8D 03 04 00 92 8D 02 0D 00 8D 02 0E 00 8D 06 16 00 3F EC FF FF 8D
02 17 00 03 00
```

The response packet is deconstructed and explained in Table 12.

Table 12: A Response Packet

Octets (Hexadecimal)	Description
84 00	A Cancel response
81 00	A Begin response
8D 06 01 00 56 34 12 00	A response for writing CSAP attribute cNodeAddress
8D 05 02 00 EF CD AB	A response for writing CSAP attribute cNetworkAddress
8D 03 03 00 0A	A response for writing CSAP attribute cNetworkChannel
8D 03 04 00 92	A response for writing CSAP attribute cNodeRole
8D 02 0D 00	A response for writing CSAP attribute cCipherKey, key not repeated in response
8D 02 0E 00	A response for writing CSAP attribute cAuthenticationKey, key not repeated in response
8D 06 16 00 3F EC FF FF	A response for writing CSAP attribute cFeatureLockBits
8D 02 17 00	A response for writing CSAP Attribute cFeatureLockKey, key not repeated in response
03 00	An End response

In order to take the settings into use, an Update request needs to be sent. An Update request with a countdown time of 180 seconds is as follows (octets shown in hexadecimal):

```
05 02 B4 00
```

The node will respond with an Update response:

```
85 02 C5 00
```

Note, that the response contains a countdown value that is up to 20 seconds greater than the value in the request. See section 3.6 for an explanation of why this happens.

After about three minutes, the node will reboot and have the settings taken into use. Since the feature lock key is set, any further configuration needs to use the Begin with Lock request with the correct key.

Sometimes it is desirable to clear the feature lock key. This can be done with the following request packet (octets shown in hexadecimal):

```
04 00 02 10 FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00 0D 12 17
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 03 00 05 02 0A 00
```

A deconstruction of the request packet is shown in Table 13.

Table 13: Another Request Packet

Octets (Hexadecimal)	Description
04 00	Issue a Cancel request, to put Remote API in a known state.
02 10 FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00	Issue a Begin with Lock request with the correct key.
0D 12 17 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	Write a blank key to CSAP Attribute cFeatureLockKey.
03 00	Issue an End request.
05 02 0A 00	Issue an Update request with the smallest possible countdown value of 10 seconds.

The node will respond (octets shown in hexadecimal):

```
84 00 82 00 8D 02 17 00 83 00 85 02 1F 00
```

A deconstruction of the response packet is shown in Table 14.

Table 14: Another Response Packet

Octets (Hexadecimal)	Description
84 00	A Cancel response
82 00	A Begin with Lock response
8D 02 17 00	A response for writing CSAP Attribute cFeatureLockKey, key not repeated in response
83 00	An End response
85 02 1F 00	An Update response, with actual countdown value returned

After the update countdown has elapsed, the node will set a blank key to cFeatureLockKey CSAP attribute, disabling the feature lock. Since cFeatureLockKey attribute does not require a reboot, the node continues operating normally after the update.

Attributes that contain bits that control the node operation, namely cNodeRole and cFeatureLockBits can be adjusted without affecting all bits at the same time. An optional bit mask was introduced in firmware v3.4.0 for these two attributes.

The following request sets the low-latency bit (bit 7) without affecting the other bits (e.g. auto-role and subnode or headnode selection) in the node role. It then clears the OTAP Disable bit (bit 31) in the cFeatureLockBits attribute. The Begin and End requests have been omitted for brevity.

```
0D 04 04 00 80 80 0D 0A 16 00 00 00 00 00 80 00 00 00
```

A deconstruction of the request packet is shown in Table 15.

Table 15: Writing Individual Bits of cNodeRole and CFeatureLockBits

Octets (Hexadecimal)	Description
0D 04 04 00 80 80	Set bit 7 in CSAP attribute cNodeRole
0D 0A 16 00 00 00 00 00 80 00 00 00	Clear bit 31 in CSAP attribute cFeatureLockBits

Responses to these requests do not have the bit mask, only the resulting value, i.e. original bits where a bit in the mask was cleared and new bits where the mask had a bit set.

5. References

[1] WP-RM-100 – Wirepas Mesh Dual-MCU API Reference Manual

[2] WP-AN-324 – Open Joining and Online Provisioning