Weightless Starter Kit
Interface with Protocol Stack

Revision History

Revision Code	Date	Description	Comments
1.0.0	Sep 5, 2017	Firmware 1.0.0	Initial Draft
1.0.5	Sep 22, 2017	Firmware 1.0.5	Added Submode to run mode. Added Parameters for Test Modes. Added “Raw Parameter” Command
1.0.7	Nov 1, 2017	Firmware 1.0.7	Added Downlink Acknowledgement and base Station Log message types
1.0.8	Jan 1, 2018	Firmware 1.0.8	Add FullUplink Add Json
1.0.9	Apr 09, 2018	Firmware 1.0.9	Added NAK Added Frequency Scan
1.0.9	May 09, 2018	Firmware 1.0.9	Changed Multicast groups message format Added device status *unknown*
1.0.10	Aug 25, 2018	1.0.10	Removed old message Uplink

Table of Contents

Revision History 1

Table of Contents 2

Introduction 6

Communication Flow 7

Communication format 7

CRC Algorithm 7

How To Connect 8

Authentication 8

Payload format 10

Message Direction 11

Message Types 12

Discovery 12

Downlink 12

Multicast Downlink 12

Command 12

Base Station Log 12

Downlink Acknowledgement 13

Command Response 13

Device Status 13

Device Status Format 14

Acknowledge 14

Negative Acknowledge (NACK) 14

JSON 14

Header Format 14

Payload Format 14

Uplink 14

Header Format 15

Payload Format 15

Multicast Group Notification 15

Header Format 15

Payload Format 15

Pong 16

Header Format 16
Payload Format 16
HardwareInfo 16
Header Format 17
Payload Format 17
KeyRequest 17
Header Format 17
Payload Format 17
KeyResponse 17
Header Format 17
Payload Format 17
RegisterRequest 17
Header Format 18
Payload Format 18
RegisterResponse 18
Header Format 18
Payload Format 18
FileFragment 18
Header Format 18
Payload Format 18
Capabilities 18
Header Format 19
Payload Format 19
FirmwareUpdateNotification 19
Header Format 19
Payload Format 19
KeepAlive 19
Header Format 19
Payload Format 20
Authenticated 20

Commands 21
Command 21
Command Types 21

Command Response 22
General Error Codes 22
Commands Format 22
Base Channel 22
1- Read Base Channel 22

Response 22
2- Set Base Channel 23
Response 23
Multicast 23
1- Read Multicast Groups 23
Response 23
2- Add Devices to Multicast Groups 23
Response 24
3- Remove Devices from Multicast Groups 24
Response 24
4- Set Multicast Groups 24
Response 24
SIB 24
1-Set SIB 24
Response 25
2- Read SIB 25
Response 26
Blacklist Channel 26
1- Blacklist N Channels 26
Response 26
2- Remove N Channels from Blacklist 26
Response 26
3- Get Blacklisted channels 26
Response 26
4- Set Blacklisted Channels 26
Response 26
Parameter 27
Set / Get parameter 27
Response 27
Currently Supported Parameters 27
Run Mode 28
Set / Get 28
Response 28
Raw Parameter 28
Response 28
Currently Supported Raw Parameters 29
Frequency Scan 30
Response 30
JSON Updates 30

Base Station Firmware Update 30
Response 31
Ping Request 31
Response 31
Query End Device Input Queue Size 31
Response 31
End Device MCS Settings (GET, SET) 31
Response 31
End Device Current MCS Value (GET, SET) 32
Response 32
End Device RA Slots Value (GET, SET) 32
Response 32
PutFile 32
Response 33
RoamEndDevice 33
Response 33
GetFile 33
Response 33
Syscall 33
Response 33
FirmwareUpdate 34
Response 34
GetFirmwareUpdateStatus 34
Response 34
GetNetworkStatistics 34
Response 34
JSON Command 34
Response 35
Command List 35
Local network Discovery via UDP 36

Introduction

Developers can reference this manual to develop their own application to connect to their Weightless Base Station without needing to be connected to the internet.

Communication Flow

From a top level view, the communication with the Protocol Stack is as follows:

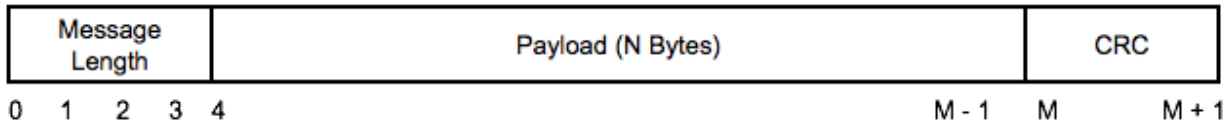
1. Connect
2. Authenticate

While connected:

3. Receive message, send ACK
4. Send message, receive ACK

Communication format

Communication with the Base Station is done through a set of messages that comply to the following format.



Byte 0-3: Length of the payload (4 Bytes) . It does not include these 4 bytes nor the 2 bytes for the CRC.
Byte 4 - (M-1): Message
Byte M - (M+1): CRC (2 bytes)

Little Endian is used across the whole system.

CRC Algorithm

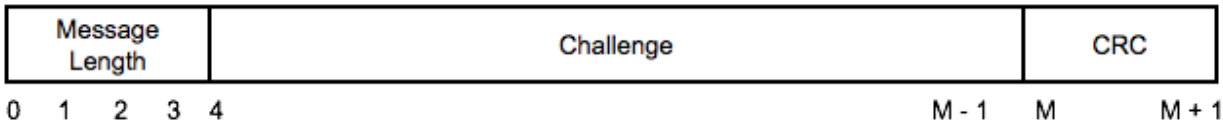
```
static UInt16_t uoi_skt_crc16(const UInt8_t* data_p, int length)
{
	UInt8_t x;
	unsigned short crc = 0xFFFF;

	while (length-- > 0) {
		x = (crc >> 8) ^ *data_p++;
		x ^= x >> 4;
		crc = (crc << 8) ^ ((UInt16_t)(x << 12)) ^ ((UInt16_t)(x << 5)) ^ ((UInt16_t)x);
	}
	return crc;
}
```

How To Connect

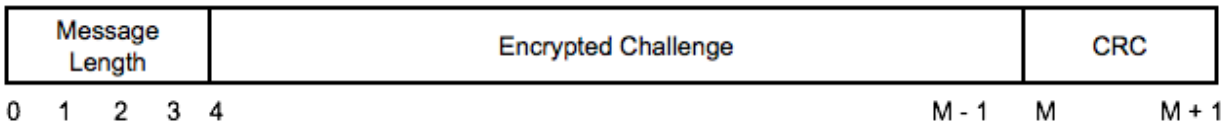
In order to connect to the Base Station to send downlinks and commands, and to receive uplinks and command responses, your application needs to connect to an application running in the Base Station that from now on will be called *Protocol Stack.*
Once the Base Station is on and Protocol Stack is running, it will listen to incoming connections on the port 7979/TCP and to discovery queries on 7979/UDP.

Authentication

When an application connects to the Protocol Stack, the Protocol Stack will send a Discovery message with a 16 byte Base Station ID, followed by a challenge. The challenge is a message with random content to the application and no message type. The application must not send an ACK for the Base Station ID Message or the challenge.



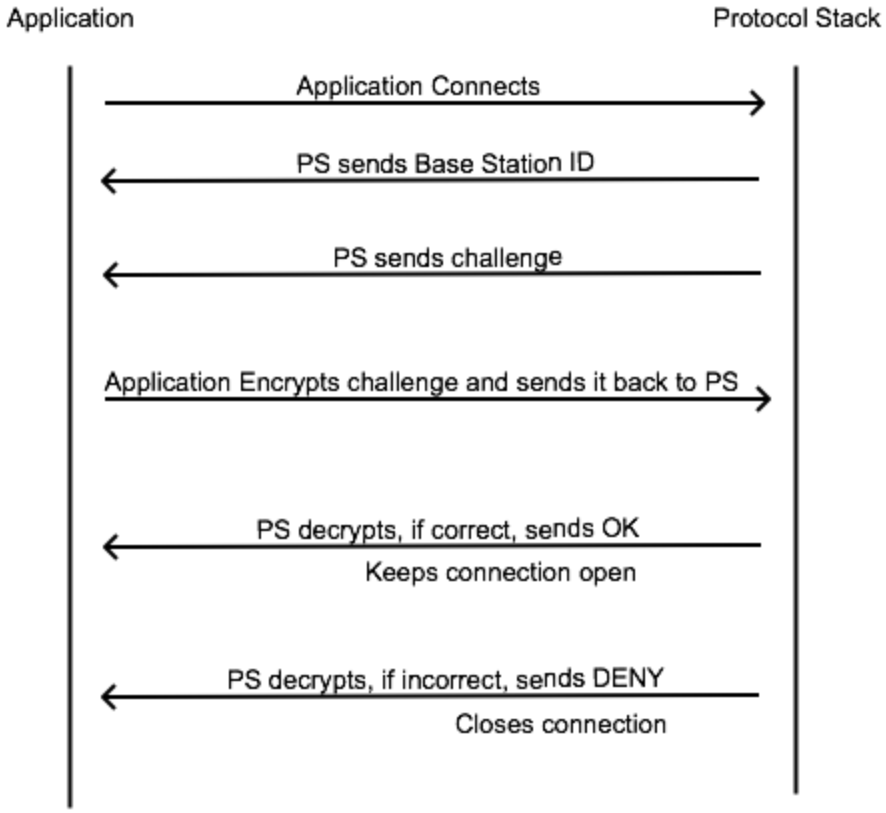
The application must encrypt the message with a shared key and send it to the protocol stack (again, no message type).



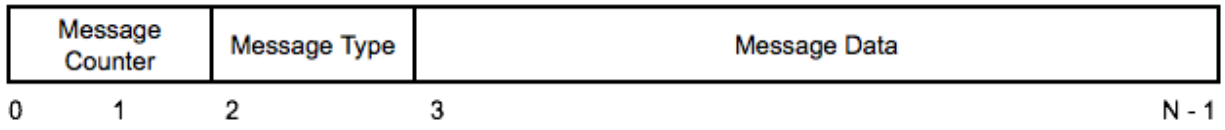
The protocol stack will check if the encrypted message matches one of results expected and will grant the application access to a series of commands. It will reply by sending a message of type “Authenticated” (0x00) and content “accepted” (0xAC).
If the message returned by the application does not match any of the expected results, the Protocol Stack will close the connection.

The encryption algorithm used is AES 128

- AES/CBC/NoPadding (128)
- IV 16 bytes


Application
Protocol Stack
Application Connects
PS sends Base Station ID
PS sends challenge
Application Encrypts challenge and sends it back to PS
PS decrypts, if correct, sends OK
Keeps connection open
PS decrypts, if incorrect, sends DENY
Closes connection

Payload format



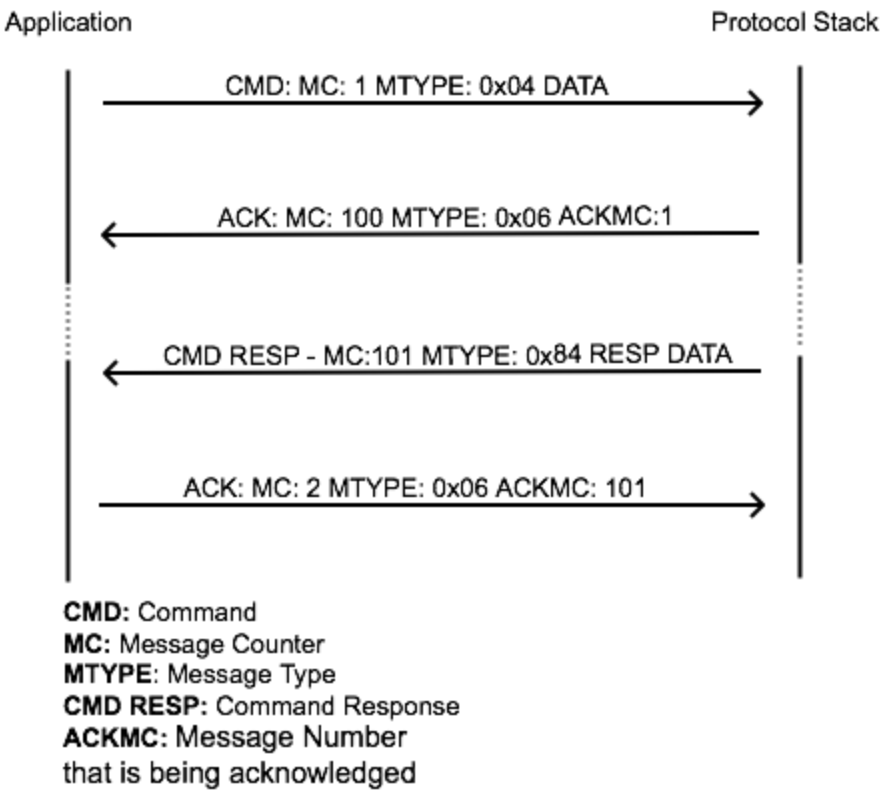
Where N = M - 4
Byte 0-1: Message Counter (2 bytes). This is a counter that increments by 1 with each message sent by the application
Byte 2: Message Type

- 0x02: Downlink
- 0x03: Multicast Downlink
- 0x04: Command
- 0x05: Devices Status
- 0x06: Acknowledge
- 0x07: Base Station Log
- 0x08: Downlink Acknowledgement
- 0x09: JSON
- 0x0A: Uplink
- 0x0B: Negative Acknowledge
- 0x0C: Multicast Notification
- 0x0D: Pong
- 0x0E: Hardware Info
- 0x7F: Discovery (Deprecated)
- 0x7C: Discovery
- 0x84: Command Response
- 0x00: Authenticated

If bit at 0x80 is 1, it means the message is a response.
Byte 3 - (N-1): message data

Protocol Stack to Application:
Uplinks
Response to Commands
Device Status
Base Station Log
Downlink Acknowledgement
Application to Protocol Stack:
Downlinks
Commands
Bidirectional:
Acknowledge

Discovery



Message Flow Example. Note that the Message Counter sent by the Protocol and the application can be different.

Each application connected to the Protocol Stack will have it's own *Message Count*. The Protocol Stack will also have it's own message count which won't necessarily match the message count in the application.

The Protocol Stack will use the Message Count to track that the messages sent to an application have been received properly. Every time the Protocol Stack receives a non ACK message, it will reply with an ACK message containing the Message Count sent by the application. The reciprocate must be implemented (except for authentication and discovery). This means that after the Protocol Stack sends a non ACK message, the application must immediately reply with an ACK whose content will be the Message Count sent by the Protocol Stack.

Message Direction

From Protocol Stack to Application will be referred to as *Up*.

From Application to Protocol Stack will be referred as *Down*.

Messages valid in both directions owuld be marked with *UpDown*

Message Types

Discovery

Message Type 0x7C: *Up*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: PayloadSize in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

Uint8: Device Type
UInt8[16]: Base Station ID

Discovery (old version, deprecated)

Message Type 0x7F: *Up*
UInt8[16]: Base Station ID

Downlink

Message Type 0x02: *Down*
UInt16: Downlink Sequence Number
UInt8: Port / Endpoint
    0x00 for unacknowledged unicast channel (LCH_UUD)
    0x01 for acknowledged unicast channel (LCH_UAD)
    0xFF means it is sending Firmware
UInt8[16]: End Device UUID
UInt8[]: Data. Specific to ED application

Multicast Downlink

Message Type 0x03: *Down*
UInt16: Downlink Counter
UInt8: Port / Endpoint (For future use).
    0x00 for un unacknowledged communication
    0x01 for acknowledged communication

0xFF means it is sending Firmware
UInt16: Multicast Group ID
UInt8[]: Data. Specific to EDs application

Command

Message Type 0x04: *Down*
UInt16: Command Counter (sequence number)
UInt16: Command Type
UInt8[]: Command data

Base Station Log

Message Type 0x07. From Protocol Stack to Application
UInt16: Source
UInt16: string length in bytes
char[]: Log string (UTF8)

Downlink Acknowledgement

Message Type 0x08: *Up*
UInt32: Downlink Sequence Number
UInt32: Status (0 is OK, others are error)

Command Response

Message Type 0x84: *Up*
UInt16: Command Counter
UInt16: Command Type
UInt8: Response Status / Error Code.
UInt8[]: Command response data
More on commands in the *Commands* section

Device Status

Message Type 0x05: *Up*
UInt8: Device Type (End Device, Base Station)
0x00: End Device
0x01: Base Station
0x02: Base Station Controller
0x64: Foreign Device //100
0x65: Foreign Base Station //101

If End Device:
UInt8[16]: End Device UUID
UInt8: Device Status
Int32: joinTime (UNIX timestamp)
Int16: RSSI
Int32: connection time (UNIX timestamp)
Int32: last activity time (UNIX timestamp) (uplink or connection)
UInt32: total unacknowledged uplinks
UInt32: total acknowledged uplinks
Int8 tx: MaxPower;
Int8 tx: Power;
Uint16: mtu;
UInt8[4]: Firmware version

If Base Station / Foreign Base Station:
UInt16: Base Station (Network) ID
UInt8: Base Station Status
UInt8[4]: Protocol Stack Version (Major, Minor0, Minor1, Minor2)
Char[16]: BuildDateTime null-terminated string (e.g. “20171231T235959”)
UInt32: number of End Devices

If Foreign End Device:
UInt8[16]: End Device UUID
UInt8: Device Status
Int32: connection time (UNIX timestamp)
UInt8[4]: Firmware version

Device Status Format

0x00 Unknown
0x01 Registering
0x02 Connected
0x03 Disconnected
0x04 NotProvisioned
0x05 NotFound
0x06 Registered
0x07 Rejected

Acknowledge

Message Type 0x06: *UpDown*

UInt16: Message Counter. The sequence number of the message that is being acknowledged

Negative Acknowledge (NACK)

Message Type 0x0B: *UpDown*
UInt16: Message Counter. The sequence number of the message that is being negatively acknowledged
UInt16: Error Code

JSON

Message Type 0x09: *UpDown*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: PayloadSize in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[PayloadSize]: JSON string

Uplink

Message Type 0x0A: *Up*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: PayloadSize in bytes (4 bytes)

Header Format

UInt32: Protocol Version
UInt64: Uplink Unique ID. Formed by

- UInt32 for sequence number (32 LSB)
- UInt32 for timestamp (32 MSB)

UInt16: Base ID
UInt16: Port / Endpoint
0x00 for un unacknowledged communication
0x01 for acknowledged communication
Int16: Uplink RSSI (in dBm)
UInt8[16]: End Device UUID

Payload Format

UInt8[PayloadSize]: Application Protocol Data Unit (APDU)

Multicast Group Notification

Message Type 0x0C: *Up*
UInt32: HeaderSize in bytes
UInt32: PayloadSize in bytes

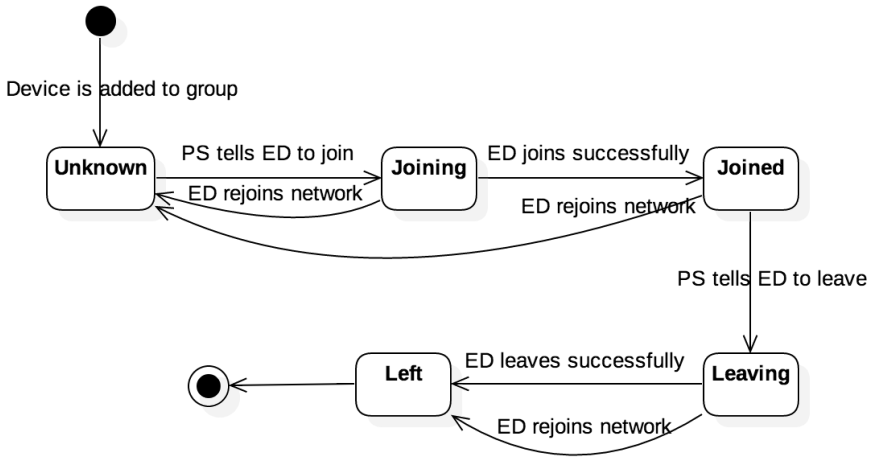
Header Format

UInt32: Protocol Version
UInt16: Number of groups
UInt8: Type
- 0: FullUpdate
- 1: Partial Update

Payload Format

UInt8[PayloadSize]: Application Protocol Data Unit (APDU)

For each group:
UInt16: Multicast Group ID
UInt16: Number of devices in group
For each device:
- **UInt8**[16]: UUEID
- **UInt8**: EdStatus
  - 0: Unknown
  - 1: Joining
  - 2: Joined
  - 3: Leaving
  - 4: Left



State Diagram for End Device in Multicast Group

Pong

Message Type 0x0D: *Up*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: PayloadSize in bytes (4 bytes)

Header Format

UInt32: Protocol Version
UInt8[16]: End Device UUEID
UInt32: Ping ID

Payload Format

UInt32: Timestamp
UInt16: Data size in bytes
Int32: Roundtrip delay in milliseconds (<= 0 is error)

HardwareInfo

Message Type 0x0E: *Up*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[N]: Hardware Hash

KeyRequest

Message Type 0x0F. *Up*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[16]: UUEID

KeyResponse

Message Type 0x10. *Down*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[16]: UUEID
UInt8[16]: Kmaster

RegisterRequest

Message Type 0x11. *Up*..
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[16]: UUEID

RegisterResponse

Message Type 0x12. *Down*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version

Payload Format

UInt8[16]: UUEID
Uint32: result //1: accept 0: reject

FileFragment

Message Type 0x13. *Up*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version
UInt32; fileId
UInt32: fragSize
UInt64: fragOffset

Payload Format

UInt8[N]: Fragment Data

Capabilities

Message Type 0x14. *Up*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version
UInt8[16]: UUEID

Payload Format

Int8: txMaxPower (in dBm)
Int8: txPower (in dBm)
UInt16: MTU (Maximum Transmission Unit size in bytes)
UInt8[4]: Firmware Version

FirmwareUpdateNotification

Message Type 0x15. *Up*.
UInt32: HeaderSize in bytes (4 bytes)
UInt32: Payload Size (N) in bytes (4 bytes)

Header Format

UInt32: Protocol Version
UInt32: numDevices

Payload Format

struct **FwuStatus**
{
    UInt8[16]: UUEID
    UInt32 imageIndex
    UInt32 status: current status
    UInt32 progress: progress indicator
}
FwuStatus[numDevices]: Array of statuses for the numDevices devices

KeepAlive

Message Type 0x7D: *Down*
UInt32: HeaderSize in bytes (4 bytes)
UInt32: PayloadSize in bytes (4 bytes)

Header Format

Int32: Current time (UNIX timestamp)

Payload Format

UInt8[]: Ignored

Authenticated

UInt8: Authentication Status
0xAC: accepted

Commands

Commands and responses are asynchronous and there is no guarantee on how long it will take for a command to be processed.
When you send a command to the Protocol Stack, you won't get a response immediately. You will only get an ACK and some time later the response.
It is the application's responsibility to implement a timeout mechanism.

Command

Message Type 0x04: *Down*
UInt16: Sequence Number
UInt16: Command Type
UInt8[]: Command data

Command Types

The following is the list of commands Available. All commands undocumented should not be used.

0x01 Read Base Frequency
0x02 Set Base Frequency
0x03 Read Multicast Groups
0x04 Add Devices to Multicast Groups
0x05 Remove Devices from Multicast Group
0x06 Set Multicast Groups
0x07 Blacklist N Channels
0x08 Remove N Channels from Blacklist
0x09 Get Blacklisted channels
0x0A Set Blacklisted Channels
0x0D Set SIB
0x0E Read SIB
0x10 Parameter
0x14 Run Mode
0x15 Raw Parameter
0x17 Set Current Configuration as Default
0x18 Frequency Scan
0x19 Base Station Firmware Update
0x20 Ping Request
0x21 Query End Device Input Queue Size
0x22 MCS Settings

0x23 MCS Value

0x30 PutFile
0x31 RoamEndDevice
0x32 GetFile
0x33 Syscall
0x34 FirmwareUpdate
0x35 GetFirmwareUpdateStatus
0x36 GetNetworkStatistics

0xFE JSON command

Command Response

Message Type 0x84. *Up*.
UInt16: Sequence Number
UInt16: Command Type
UInt8: Response Status / Error Code.
UInt8[]: Command response data

General Error Codes

0x00: Success
0x01: Unknown command (This error will be returned if the application doesn't have permission to execute the command).
0xFF: Unknown Error
Other error codes depend on the command type

Commands Format

Base Channel

Base Channel is in WARFCN

1- Read Base Channel

Command Type 0x01
No Parameters

Response

UInt16: Channel

2- Set Base Channel

Command Type 0x02
UInt16: Channel
Valid Channels [4000, 5100], [8000, 10200]

Response

Response has no content

Specific error codes:
0x02: Invalid Channel

Multicast

1- Read Multicast Groups

Command Type 0x03

Response

UInt16: number of multicast groups (N)

For each group:
UInt16: Multicast Group ID
UInt16: Number of devices in group (M)
For each Device
    UInt8[16]: UUIDs of End Devices
    UInt8: EdStatus
        0: Unknown
        1: Joining
        2: Joined
        3: Leaving
        4: Left

2- Add Devices to Multicast Groups

Adds devices to a multicast group. If the group doesn't exist, it will create it.
Command Type 0x04.
UInt16: Number of Multicast Groups
Followed by N groups of:
UInt16: Multicast Group ID
UInt16: number of devices in group (M)
UInt8[M][16]: UUIDs of End Devices

Response

Response has no content

3- Remove Devices from Multicast Groups

Command Type 0x05.
UInt16: Number of Multicast Groups
Followed by N groups of:
UInt16: Multicast Group ID
UInt16: number of devices in group (M)
UInt8[M][16]: UUIDs of End Devices

Response

UInt8: Status

4- Set Multicast Groups

Command Type 0x06.
UInt16: Number of Multicast Groups
Followed by N groups of:
UInt16: Multicast Group ID
UInt16: number of devices in group (M)
UInt8[M][16]: UUIDs of End Devices

Response

UInt8: Status

SIB

1-Set SIB

Command Type 0x0D.

UInt8: FRAME_DURATION
value: 0 ~ 3
Description:
0: 2 seconds
1: 4 seconds
2: 8 seconds
3: 16 seconds

Byte 1:

For future use. Set to 0xFF

Byte 2:
For future use. Set to 0xFF

Byte 3:
For future use. Set to 0xFF

UInt8: SIB1_PRESENT
value: 0 or 1
description:
0: the following hopping setting is absent
1: the following hopping setting is present

If SIB1_PRESENT is not 0, then the following are included:

UInt16: HOP_FIRST_CH (WARFCN)
value: 0 ~ 10000
description: first channel number in hop sequence

UInt8: HOP_CH_SPACING
value: 0 ~ 255
description: hop channel spacing in 100kHz unit (0 = 100kHz, 1: 200kHz…)

UInt8: HOP_CH_NB
value: 0 ~ 63
description: number of hopping channels minus 1

UInt8: NB_NCELL_CH
value: 0 ~ 6
description: number of inter-frequency neighbor-cell channel

UInt16[*NB_NCELL_CH]*: NCELL_CH (WARFCN)
value: 0 ~ 10000
description: inter-frequency neighbor-cell channels

Response

Response has no specific content

2- Read SIB

Command Type 0x0E.

Response

Same as for Set SIB Command

Blacklist Channel

1- Blacklist **N** Channels

Command Type 0x07.
UInt16: Number of channels (N)
UInt16: Channels

Response

Specific error codes:
0x02: Invalid Channel

2- Remove N Channels from Blacklist

Command Type 0x08.
UInt16: Number of channels (N)
UInt16: Channels

Response

Specific error codes:
0x02: Invalid Channel

3- Get Blacklisted channels

Command Type 0x09.

Response

Command Type 0x09
UInt16: Number of channels (N)
UInt16: Channels

4- Set Blacklisted Channels

Command Type 0x0A.
UInt16: Number of channels (**N**)
UInt16[N]: Channels

Response

Command Type 0x0A
Specific error codes:

0x02: Invalid Channel

Parameter

Set / Get parameter

Command type 0x10
Char[32]: command name (ASCII String 0 padded)
UInt16: Index
UInt8: Operation Type (0: GET, 1: SET)
Int32 ~~UInt32~~: Value (Only if operation is SET.)

Response

Int32 ~~UInt32~~: Value (only if operation is GET)

Currently Supported Parameters

name	index	min	max	default	description
NBFLAG	0	0	1	0	0: wide band is used in UL 1: narrow band is used in UL
SIB_MCS	0	0	3	0	0: GMSK, 100Kbps, FEC OFF 1: PSK, 12.5Kbps, FEC OFF 2: GMSK, 50Kbps, FEC ON 3: PSK, 6.25Kbps, FEC ON
UL_MCS	0	0	3	0	0: GMSK, 100Kbps, FEC OFF 1: PSK, 12.5Kbps, FEC OFF 2: GMSK, 50Kbps, FEC ON 3: PSK, 6.25Kbps, FEC ON
UL_MCS_NB	0	0	3	0	0: GMSK, 10Kbps, FEC OFF 1: PSK, 1.25Kbps, FEC OFF 2: GMSK, 5Kbps, FEC ON 3: PSK, 0.625Kbps, FEC ON
ED_MCS_CTRL	0	0	1	0	0: adaptive MCS control is disabled 1: adaptive MCS control is enabled
ED_TX_PWR_CTRL	0	0	1	0	0: adaptive power control is disabled 1: adaptive power control is enabled
FOR TEST MODES					
UT_FREQ	0	400000	1020000	undef	Test carrier frequency in kHz [400000, 510000], [800000, 1020000]

UT_TIME_MS	0	0	Max int	undef	Test time in milliseconds. 0 will make the test go on continuously and must be stopped manually
UT_TX_PWR	0	-17	14	undef	ut_tx_pwr (tx test transmit power: 31 corresponding to 14dBm, 1dB per step, valid range is 0~31)

Run Mode

With this command you can read the current running mode of the Base Station or Set the desired mode

Set / Get

Command type 0x14
UInt8: Operation Type
- 0: GET
- 1: SET

If SET:
UInt16: Mode
- 0: Stop
- 1: Start
- >1: Test Modes

UInt16: Submode
- For future use, must be 0

Response

(only if operation is GET)
UInt16: Mode (0xFFFF if Initial State)
UInt16: Submode

Raw Parameter

Command type 0x15
Char[32]: Parameter name (ASCII String 0 padded)
UInt8: Operation Type (0: GET, 1: SET)
UInt16: Content length N (0 if GET)
UInt8[N]: Raw parameter value

Response

UInt16: Content length N
UInt8[N]: Response content

Currently Supported Raw Parameters

Parameter	GET	SET	Description	Example
TEST_MODES	Yes	No	JSON description of available test modes	{ "test_modes" : [{ "category":"rx", "tests": [{ "name": "GFSK 100kbps", "run_mode": 2, "sub_mode":0 }, { "name": "GFSK 50kbps", "run_mode": 3, "sub_mode":0 }] }, { "category":"tx", "tests": [{ "name": "CW", "run_mode": 4, "sub_mode":0 }, { "name": "mcs all", "run_mode": 5, "sub_mode":0 }] }] }

Frequency Scan

Command type 0x18
UInt16: Start WARFCN
UInt16: Stop WARFCN
UInt16: WARFCN step (min 1)
UInt16: Time Window (milliseconds)
UInt16: Number of windows

Response

Empty

Error Codes:
Invalid Start WARFCN 0x02
Invalid Stop WARFCN 0x04
Invalid WARFCN step (min 1) 0x08
Invalid Time Window (milliseconds) 0x10
Invalid Number of windows 0x20

JSON Updates

After the command is executed successfully, the Protocol Stack will send the scan results periodically using a JSON message (message Type 0x09).

```
{
“frequency_scan”:
  {
    “samples”:
    {
      “warfcn”:[4500,4550,...],
      “min”:[-50,-51,...],
      “max”:[-60,-59,...],
      “avg”:[-53,-54,...],
    }
  }
}
```

Base Station Firmware Update

Command type 0x19.
UInt32: File Length FL
UInt8[FL]: File

Response

Command type 0x19
UInt32: Firmware upgrade result/error code

Ping Request

Command type 0x20
UInt8[16]: UUEID
UInt32: Ping ID
UInt16: Ping payload size in bytes

Response

Empty

Query End Device Input Queue Size

Command type 0x21.
UInt8[16]: UUEID

Response

UInt32: Number of messages in LCH_UUD input queue
UInt32: Total size in bytes in LCH_UUD input queue
UInt32: Number of messages in LCH_UAD input queue
UInt32: Total size in bytes in LCH_UAD input queue

End Device MCS Settings (GET, SET)

Command type 0x22
UInt8: Operation Type (0: GET, 1: SET)
UInt8[16]: UUEID

If operation type is SET:
UInt16: Downlink MCS bitmap
UInt16: Uplink MCS bitmap (WB and NB)

1C0022000103A18D4BDCEC292E0A188F4AB2A2F272
0300
3300

Response

Command type 0x22
If operation type is GET:
UInt16: Downlink MCS bitmap
UInt16: Uplink MCS bitmap(WB and NB)

End Device Current MCS Value (GET, SET)

Command type 0x23
UInt8: Operation Type (0: GET, 1: SET)
UInt8[16]: UUEID

If operation type is SET:
UInt8: Current Downlink MCS Value
UInt8: Current Uplink MCS Value
UInt8: Current Uplink MCS Value (Narrowband)

Response

If operation type is GET:
UInt8: Current Downlink MCS Value
UInt8: Current Uplink MCS Value
UInt8: Current Uplink MCS Value (Narrowband)

End Device RA Slots Value (GET, SET)

Command type 0x24
UInt8: Operation Type (0: GET, 1: SET)
UInt8[16]: UUEID

If operation type is SET:
UInt16: Number of RA slots
UInt16: Number of RA Slots (slow MCS)

Response

If operation type is GET:
UInt16: Number of RA slots
UInt16: Number of RA Slots (slow MCS)

PutFile

Command type 0x30
Char[256]: filename
UInt32: fragOffset
UInt32: fragSize
UInt8[fragSize]: Fragment data

Response

Empty

RoamEndDevice

Command type 0x31
Uint8[16]: UUEID
UInt32: reason

Response

Empty

GetFile

Command type 0x32
UInt32: fragSize
Char[]: filename

Response

UInt32: fileId (for subsequent FileFragment messages)
UInt64: fileSize (in bytes)
UInt8[32]: SHA256 hash

Syscall

Command type 0x33
Int32: timeout (in seconds)
UInt32: cmdLnegth (in characters)
Char[cmdLength]: command

Response

Int32: returnValue
UInt32: outputLength
Char[outputLength}: output (captured from command standard output)

FirmwareUpdate

Command type 0x34
Char[256]: firmwareFilename
Char[256]: previousFirmwareFilename: set to '\0' to disable incremental firmware update
UInt32: numDevices
UInt32: imageIndex: 0 for main firmware, others for additional images
UInt8: ackMode
UInt8 autoApply: non-zero to apply the firmware upon successful transfer
UInt8[16][numDevices]: UUEID list

Response

Empty

GetFirmwareUpdateStatus

Command type 0x35
UIn328 imageIndex
UInt32: numDevices
UInt8[16][numDevices]: UUEID list

Response

The actual status for all devices will be sent over a FirmwareUpdateNotification message

GetNetworkStatistics

Command type 0x36
UInt32: numDevices
UInt8[16][numDevices]: UUEID list

Response

To query the Base Station network statistics, use numDevices = 0.
Network statistics are sent using a JSON message with root "NetworkStatistics".

JSON Command

Command type 0xFE
UInt32: PayloadSize in bytes)
UInt8[PayloadSize]: JSON string

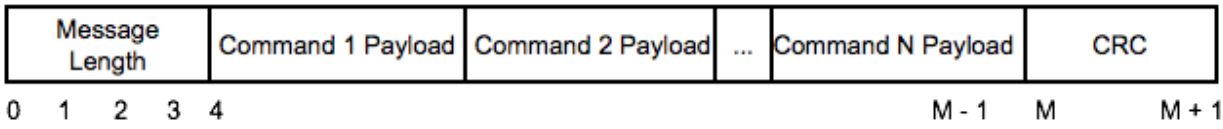
Response

UInt32: PayloadSize in bytes
UInt8[PayloadSize]: JSON string

Command List

From Application to Protocol Stack
Contains a list of commands.

For each command, it includes:
UInt16: Sequence Number
UInt16: Command Type
UInt8[]: Command data



Local network Discovery via UDP

To discover Base Stations on your local network, you should send an UDP broadcast message of type *Discovery* on port 7979
Base Station ID must be set to the 16-byte mask:
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Base station will send a message also with type *Discovery* carrying its 16-byte UUID.

END