Weightless-P Starter Kit
Interface with Protocol Stack

Revision History

Revision Code	Date	Description	Comments
1.0.0	Sept 5 2017	Initial Draft	

Table of Contents

Introduction

Users can reference the sample code in this manual to develop their own application to connect to their Weightless-P 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:

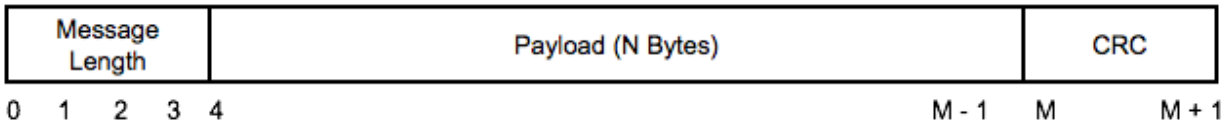
1. Connect
2. Authenticate

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

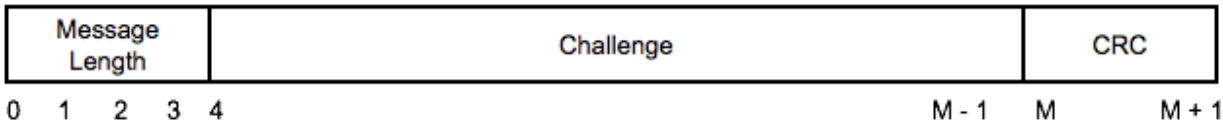
```
static uint16_t uoi_skt_crc16(const uint8_t* data_p, int length)
{
	uint8_t x;
	unsigned short crc = 0xFFFF;

	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

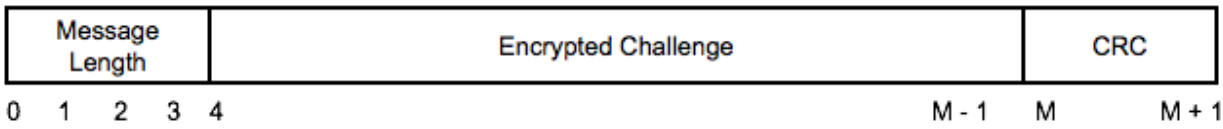
In order to connect to the Base Station to send downlinks and commands, and to receive uplinks and command responses, your application needs to connect to an application running in the Base Station that from now on will be called *Protocol Stack.*
Once the Base Station is on and Protocol Stack is running,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

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.



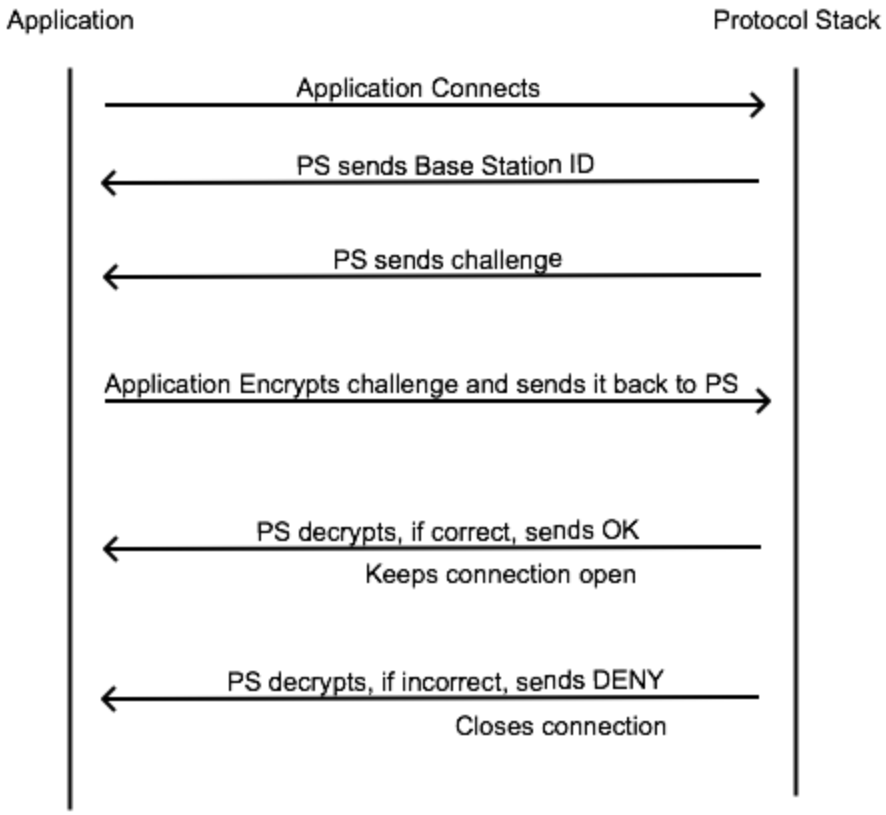
The application must encrypt the message with a shared key and send it to the protocol stack (again, no message type).



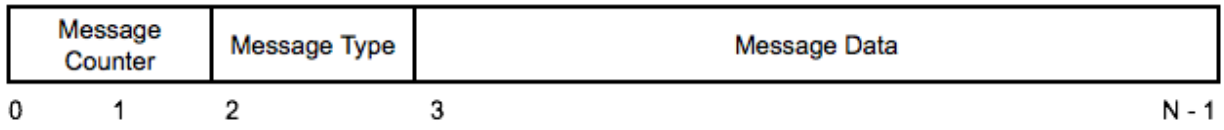
The protocol stack will check if the encrypted message matches one of results expected and will grant the application access to a series of commands. It will reply by sending a message of type “Authenticated” (0x00) and content “accepted” (0xAC).
If the message returned by the application does not match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

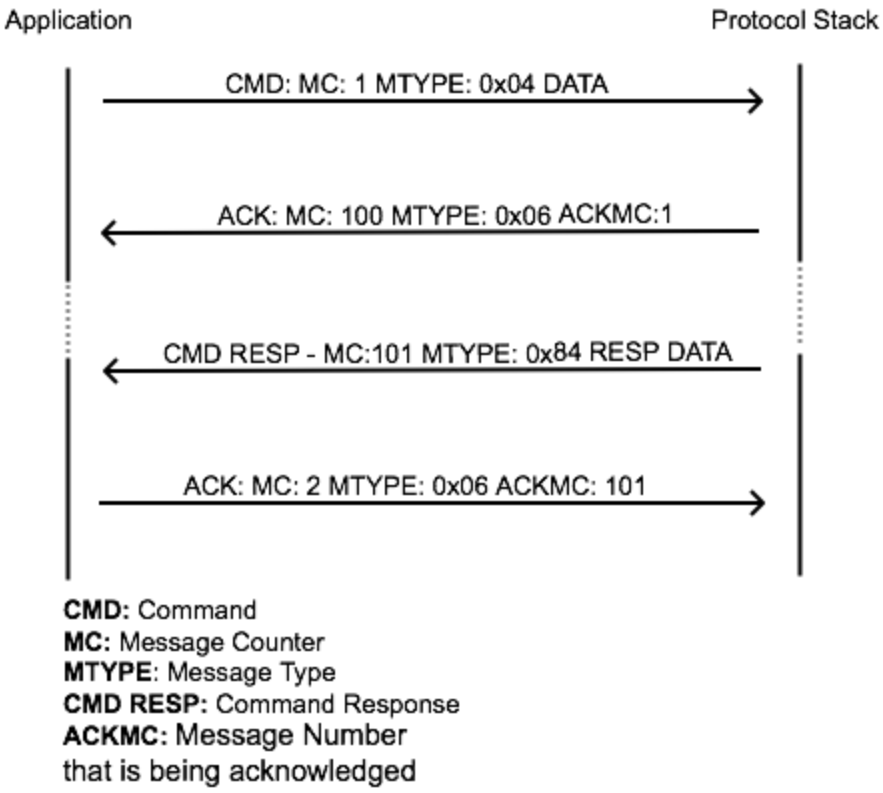


Where N = M - 4
Byte 0-1: Message Counter (2 bytes). This is a counter that increments by 1 with each message sent by the application
Byte 2: Message Type

- 0x01: Uplink
- 0x02: Downlink
- 0x03: Multicast Downlink
- 0x04: Command
- 0x05: Devices Status
- 0x06: Acknowledge
- 0x7F: Discovery
- 0x84: Command Response
- 0x00: Authenticated

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

Protocol Stack to Application:
Uplinks
Response to Commands
Device Status
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 Types

Discovery

Message Type 0x7F: Discovery. From Protocol Stack to Application.
Byte 0 - 16: Base Station ID (16 bytes)

Uplink

Message Type 0x01. From Protocol Stack to Application.
Byte 0-1: Uplink Counter (2 bytes)
Byte 2-3: Base ID (2 bytes)
Byte 4: Port / Endpoint (1 Byte, For future use)
    0x00 for un unacknowledged communication
    0x01 for acknowledged communication
Byte 5: End Device UUID (16 bytes)
Byte 21 - N: Data. Specific to EDs application

Downlink

Message Type 0x02. From Application to Protocol Stack
Byte 0-1: Downlink Counter
Byte 2: Port / Endpoint (1 Byte).
    0x00 for un unacknowledged communication
    0x01 for acknowledged communication
    0xFF means it is sending Firmware
Byte 3: End Device UUID (16 bytes)
Byte 19 - N: Data. Specific to ED application

Multicast Downlink

Message Type 0x03. From Application to Protocol Stack
Byte 0 -1: Downlink Counter
Byte 2: Port / Endpoint (For future use).
    0x00 for un unacknowledged communication
    0x01 for acknowledged communication
    0xFF means it is sending Firmware
Byte 3 - 4: Multicast Group ID (2 bytes)
Byte 5 - N: Data. Specific to EDs application

Command

Message Type 0x04. From Application to Protocol Stack
Byte 0-1: Command Counter (sequence number)
Byte 2-3: Command Type
Byte 4 - N: Command data

Command Response

Message Type 0x84. From Protocol Stack to Application
Byte 0-1: Command Counter
Byte 2-3: Command Type
Byte 4: Response Status / Error Code.
Byte 4 - N: Command response data
More on commands in the *Commands* section

Device Status

Message Type 0x05: From Protocol Stack to Application
Byte 0: Device Type (End Device, Base Station)
    0x00: End Device
    0x01: Base Station

If End Device:
Byte 1 - 17: End Device UUID
Byte 18: Device Status

If Base station:
Byte 1 - 2: Base Station (Network) ID
Byte 3: Base Station Status

Device Status Format

0x01 Registered / Joined network
0x02 Connected
0x03 Disconnected

Acknowledge

Message Type 0x06: From Protocol Stack to Application or vice versa.
Byte 0 - 1: Message Counter. The number of message that is being acknowledged

Authenticated

Byte 0: Authentication Status (1 byte)

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. From Application to Protocol Stack
Byte 0-1: Sequence Number
Byte 2-3: Command Type
Byte 4 - N: 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

Command Response

Message Type 0x84. From Application to Protocol Stack
Byte 0-1: Sequence Number
Byte 2-3: Command Type
Byte 4: Response Status / Error Code.

Byte 5 - N: 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. From Application to Protocol Stack.
No Parameters

Response

Command Type 0x01. From Protocol Stack to Application.
Byte 0 - 1: Channel (2 bytes)

2- Set Base Channel

Command Type 0x02. From Application to Protocol Stack.
Byte 0 - 1: Channel (2 bytes)
Valid Channels [4000, 5100], [8000, 10200]

Response

Command Type 0x02. From Protocol Stack to Application.
Response has no content

Specific error codes:
0x02: Invalid Channel

Multicast

1- Read Multicast Groups

Command Type 0x03. From Application to Protocol Stack.

Response

Command Type 0x03. From Protocol Stack to Application.
Byte 0 - 1: number of multicast groups (N) (2 bytes)

Followed by N groups of:
Byte 0 - 1: Multicast Group ID (2 bytes)
Byte 2 - 3: number of devices in group (M) (2 bytes)
Byte 4 - (4 + 16*M): UUIDs of End Devices (16 bytes blocks)

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. From Application to Protocol Stack.
Byte 0 - 1: Number of Multicast Groups (2 bytes)
Followed by N groups of:
Byte 0 - 1: Multicast Group ID (2 bytes)
Byte 2 - 3: number of devices in group (M) (2 bytes)
Byte 4 - (4 + 16*M): UUIDs of End Devices (16 bytes blocks)

Response

Command Type 0x04. From Protocol Stack to Application.
Response has no content

3- Remove Devices from Multicast Groups

Command Type 0x05. From Application to Protocol Stack.
Byte 0 - 1: Number of Multicast Groups (2 bytes)
Followed by N groups of:
Byte 0 - 1: Multicast Group ID (2 bytes)
Byte 2 - 3: number of devices in group (M) (2 bytes)
Byte 4 - (4 + 16*M): UUIDs of End Devices (16 bytes blocks)

Response

Command Type 0x05. From Protocol Stack to Application.
Byte 0: Status (1 Byte)

4- Set Multicast Groups

Command Type 0x06. From Application to Protocol Stack.
Byte 0 - 1: Number of Multicast Groups (2 bytes)
Followed by N groups of:
Byte 0 - 1: Multicast Group ID (2 bytes)

Byte 2 - 3: number of devices in group (M) (2 bytes)
Byte 4 - (4 + 16*M): UUIDs of End Devices (16 bytes blocks)

Response

Command Type 0x06. From Protocol Stack to Application.
Byte 0: Status (1 Byte)

SIB

1-Set SIB

Command Type 0x0D. From Application to Protocol Stack.

Byte 0: FRAME_DURATION (1 byte)
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

Byte 4: SIB1_PRESENT (1 byte)
value: 0 or 1
description:
0: the following hopping setting is absent
1: the following hopping setting is present

If Byte 4 (SIB1_PRESENT) is not 0, then the following bytes are included:

Byte 5-6: HOP_FIRST_CH (2 bytes, WARFCN)
value: 0 ~ 10000
description: first channel number in hop sequence

Byte 7: HOP_CH_SPACING (1 byte)

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

Byte 8: HOP_CH_NB (1 byte)
value: 0 ~ 63
description: number of hopping channels minus 1

Byte 9 NB_NCELL_CH (1 byte)
value: 0 ~ 6
description: number of inter-frequency neighbor-cell channel

Byte 10 to (10 + *2* * *NB_NCELL_CH* - *1*): NCELL_CH (*2* bytes WARFCN each)
type: uint16_t
value: 0 ~ 10000
description: inter-frequency neighbor-cell channels

Response

Command Type 0x0D. From Protocol Stack to Application.
Response has no specific content

2- Read SIB

Command Type 0x0E. From Application to Protocol Stack.

Response

Command Type 0x0E. From Protocol Stack to Application.

Same as for Set SIB Command

Blacklist Channel

1- Blacklist **N** Channels

Command Type 0x07. From Application to Protocol Stack.
Byte 0 - 1: Number of channels (N)
Byte 2 - 3: Channel (2 bytes per channel). Repeat N times

Response

Command Type 0x07. From Protocol Stack to Application.
Specific error codes:
0x02: Invalid Channel

2- Remove N Channels from Blacklist

Command Type 0x08. From Application to Protocol Stack.
Byte 0 - 1: Number of channels (N)
Byte 2 - 3: Channel (2 bytes per channel). Repeat N times

Response

Command Type 0x08. From Protocol Stack to Application.
Specific error codes:
0x02: Invalid Channel

3- Get Blacklisted channels

Command Type 0x09. From Application to Protocol Stack.

Response

Command Type 0x09. From Protocol Stack to Application.
Byte 0 - 1: Number of channels (N)
Byte 2 - 3: Channel (2 bytes per channel). Repeat N times

4- Set Blacklisted Channels

Command Type 0x0A. From Application to Protocol Stack.
Byte 0 - 1: Number of channels (**N**)
Byte 2 - 3: Channel (2 bytes per channel). Repeat N times

Response

Command Type 0x0A. From Protocol Stack to Application.
Specific error codes:
0x02: Invalid Channel

Parameter

Set / Get parameter

Command type 0x10. From Application to Protocol Stack
Byte 0-31: command name (ASCII String 0 padded)
Byte 32-33: Index
Byte 34: Operation Type (0: GET, 1: SET)
Byte 35-38: Value (4 bytes) Only if operation is SET.

Response

Command type 0x10. From Protocol Stack to Application

Byte 0-4: Value (4 bytes) Only if operation is GET

Run Mode

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

Set / Get Run Mode

Command type 0x14
Byte 0: Operation Type

- 0: GET
- 1: SET

Byte 1-2: Mode

- 0: Stop
- 1: Start (Restart)

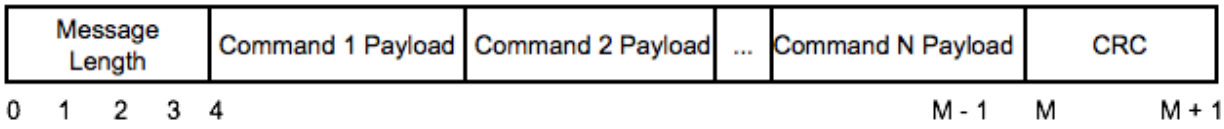
Response

Command type 0x14. From Protocol Stack to Application
Byte 0-1: Mode (2 bytes) Only if operation is GET

Command List

From Application to Protocol Stack
Contains a list of commands.

For each command, it includes:
Byte 0-1: Sequence Number
Byte 2-3: Command Type
Byte 4 - N: Command data



Local network Discovery via UDP

To discovery Base Stations on your local network, you should send an UDP broadcast message of type *Discovery*.
Base Station ID must be set to the 16 bytes mask:
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Base station will send a message also with type *Discovery* carrying it's real 16 bytes ID.

END