LWC SERVER

USER GUIDE

NOTICE

Nke Watteco reserves the right to make changes to specifications and product descriptions or to discontinue any product or service without notice. Except as provided in Nke Watteco's Standard Terms and Conditions of Sale for products, Nke Watteco makes no warranty, representation or guarantee regarding the suitability of its products for any particular application nor does Nke Watteco assume any liability arising out of the application or use of any product and specifically disclaims any and all liability, including consequential or incidental damages.

Certain applications using semiconductor products may involve potential risks of death, personal injury or severe property or environmental damage. Nke Watteco products are not designed, authorized or warranted to be suitable for use in life saving or life support devices or systems. Inclusion of Nke Watteco products in such applications is understood to be fully at the Customer's risk.

In order to minimize risks associated with the customer's application, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

Nke Watteco assumes no liability for applications assistance or customer product design. Nke Watteco does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of Nke Watteco covering or relating to any combination, machine or process in which such semiconductor products or services might be or are used. Nke Watteco's publication of information regarding any third party's products or services does not constitute Nke Watteco's approval, warranty and endorsement thereof.

Resale of Nke Watteco's products with statements of functionality different from or beyond the parameters stated by Nke Watteco for that product as defined by Nke Watteco's unique part number, voids all express and any implied warranties for that product, is considered by Nke Watteco to be an unfair and deceptive business practice and Nke Watteco is not responsible nor liable for any such use.

Embedded software is based on Nke Watteco proprietary drivers and applicative code and operates on the Contiki kernel from the SICS (Swedish Institute of Computer Science).

www.watteco.com

www.nke-electronics.com

DOCUMENT HISTORY

Date	Revision	Modification Description
July 2016	1.0	First revision
March 2017	1.1	Correcting the SF limit
March 2017	1.2	Adding the OTAA association
September 2017	1.3	Change ZCL to APP and ZIP to AIP, add packet buffer reset cmd, add PBFE error, change for downlink INDIC and add WHT resource

CONTENTS

1 INTRODUCTION

This document describes all the commands that can be used with the LWC Server (or LoRa USB Dongle) through the serial port.

In this document can be found the general way of functioning for the LWC Server, the meaning of the leds colors and the serial port configuration needed by the LWC Server to work correctly.

Then, it will describe the common syntax for the commands (special characters, operators, errors, etc.)

And finally, a complete list of all the available resources will be done with, for each, the command available and few examples.

2 DEFINITIONS

"Port" or "Slot":	Index identifying a sensor from 1 to 100
HOST:	Computer or any other device hosting and controlling the LWC Server
LWC:	LoRaWAN Server, this is the device described in this document. It can be called LoRa USB Dongle as well

3 OPERATION PRINCIPLE

The LWC server allows to create an interface between a Host (Computer, Box…) and several LoRaWAN compatible end-devices in a private network.

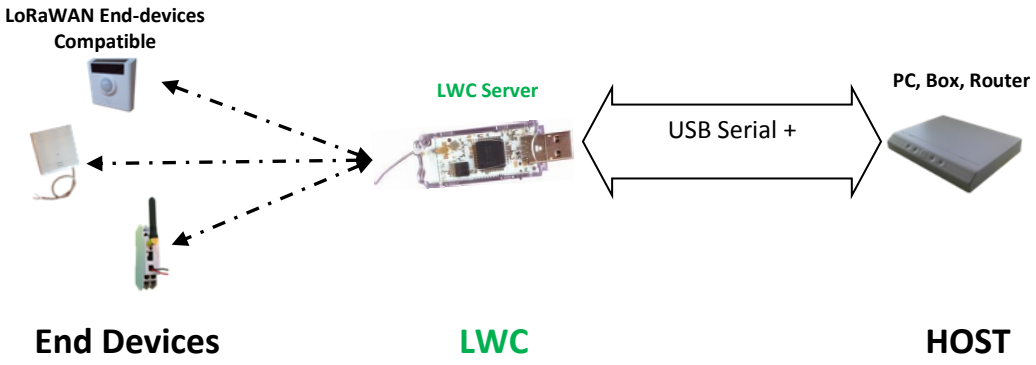

FIGURE 1 - OPERATION PRINCIPLE SCHEMATIC

4 LEDS INTERFACE

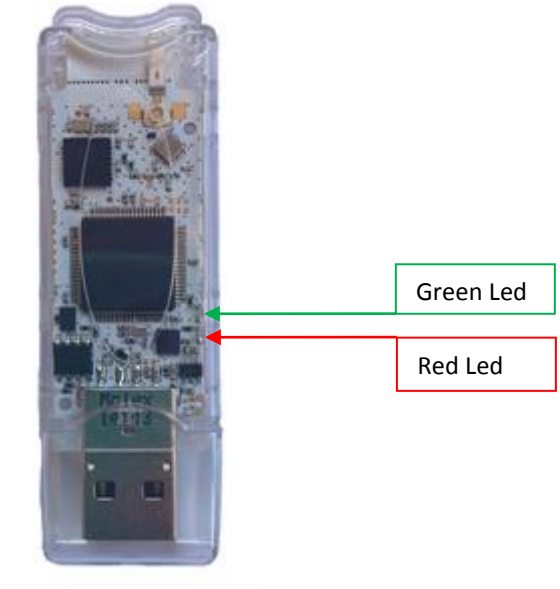

FIGURE 2 - LWC SERVER PICTURE WITH LEDS EMPLACEMENTS

Led	Meaning
	The LWC Server is ON, when the green led is continuously ON.
	The LWC Server either sends or receives a frame when the red led flashes.

5 SERIAL PORT CONFIGURATION

The LWC Server has to be plugged on a USB port on the HOST machine. The USB/serial is done through the FT232X USB (PID/VID: 0x0403/0x6015).

The serial port configuration is the following:

Baud rate: **115200 bps**
Data bits: **8 bits**
Stop bit: **1**
Flow control: **none**

6 COMMAND INTERPRETER

6.1 EXCHANGES DESCRIPTION

The exchange protocol used for the LoRaWAN Coordinator Command Line Interpreter (LWCCLI) is built with ASCII frame made up of 7 bits characters ended with the <EOL> character.

It allows 2 types of simple exchanges:

The "Request/Response":

HOST → <Request> → LWC
HOST ← <Response> ← LWC

The "Indications":

HOST ← <Indication> ← LWC

The responses can either directly return the required data or constitute a simple acknowledgement to the request. The requested data may be sent later as an <Indication>. Typically, this is the way of working for frames sent to the end-devices.

A general representation of each 3 possible command lines can be seen here below:

<Request> ::=

<Cmd><MS><DevRef>[<Param><Op><Val>]**<EOL>**

<Response> ::=

<ResType><Cmd><MS><DevRef><MS><status>[<MS><Param>[<Oper><Val>]]**<EOL>**

<Indic> ::=

<Ind><MS><DevRef>[<MS><Param>[<Op><Val>]]**<EOL>**

Each <KeyWord> is defined in the following chapters

N.B.

- The KeyWords accept two formats: a short one and a long one (ex: DEVICE_LIST <-> DL). Indeed, a specific LWC Server parameter allows to choose the wanted format: VERBOSE.
- All the non-numerical values has to respect the letter case

6.2 COMMON SYNTAX FOR ALL THE COMMANDS

6.2.1 SPECIAL CHARACTERS

The following characters are used in both ways: HOST -> LWC and LWC -> HOST.

<EOL>	::=	0x0D	*End Of Line. This character is specific to the end of an operation (Request, Response, Indication or Commentary)*
<RespType>	::=	'R'	*Command prefix, indicating a response*
<MS>	::=	' '	*Main delimiter character*
<LS>	::=	';'	*Parameters delimiter inside a parameters list*
<FS>	::=	','	*Field delimiter in a parameter containing several fields*
<Rmk>	::=	'#'	*Character indicating a commentary until the End Of Line*
<RmkStart>	::=	'<<<'	*String used to indicate the beginning of a commentary on several lines*
<RmkEnd>	::=	'>>>'	*String used to indicate the ending of a commentary on several lines*
<Dbg>	::=	'!'	*Character indicating the beginning of a debug line*

6.2.2 COMMANDS <CMD>

<Set>	::=	{SET|S}
<RSet>	::=	{RSET|RS}
<Get>	::=	{GET|G}
<RGet>	::=	{RGET|RG}
<Del>	::=	{DEL|D}
<RDel>	::=	{RDEL|RD}
<Ind>	::=	{INDIC|I}

6.2.3 OPERATORS <OP>

<Op>	::=	{"="}	The only one operator used

6.2.4 COMMON REPRESENTATIONS

<DevEUI>	::=	hhhhhhhhhhhhhhhh	16 hexadecimal digits representing the end-device's MAC Address (8 bytes BigEndian)
<DevAddr>	::=	hhhhhhhh	8 hexadecimal digits representing the end-device's LoRaWAN DevAddr (4 bytes BigEndian)
<Port>	::=	1..100	Port number or end-device index ⇨ "0" representing the LWC Server itself ⇨ The Index used to identify each end-device starts from 1
<DevRef>	::=	{<Port>|<DevEUI>|<DevAddr>}	End-device reference. It can be its index or slot number or one of its addresses.
<Class>	::=	{A|C}	LoRaWAN class of an end-device : can be A or C
<EDAM>	::=	{ABP|A}|{OTA|O}	Activation mode of an end-device.
<RX2DR>	::=	0..2	Datarate used by the LWC Server to answer on the RX2 window of end-devices This Datarate is included between 0 (SF12) and 2 (SF10).
<FPort>	::=	1..255	LoRaWAN Port used
<FCountUp>	::=	0..2^32	LoRaWAN FCount used by the end-device to send its frame
<NWSK>	::=	hhhhhhh..hhhhhhhh	32 hexadecimal digits representing the end-device's LoRaWAN NwkSKey
<APSK>	::=	hhhhhhh..hhhhhhhh	32 hexadecimal digits representing the end-device's LoRaWAN AppSKey
<AppEUI>	::=	hhhhhhhhhhhhhhhh	16 hexadecimal digits representing the end-device's LoRaWAN AppEUI (8 bytes BigEndian).
<AppKey>	::=	hhhhhhh..hhhhhhhh	32 hexadecimal digits representing the end-device's LoRaWAN AppKey

6.3 ERROR CODES

<StatusG> ::=

SUCC	Success	The command has been correctly executed
NIMP	Not Implemented	The command is not implemented yet
BCLI	Bad command line	A problem seems to be in the command line
UKCM	Unknown command	The command is unknown
UKPR	Unknown parameter	The command parameter is not valid
UKRS	Unknown Ressource	The specified ressources is not managed by the LWC Server
BADC	Bad command	The command is unavailable on the specified ressource
BADR	Bad reference	The reference (Port, DevEUI…) is not know by the LWC Server
BADP	Bad port	The specified port is invalid: it does not appear in the list of paired DevEUI. It is an empty slot or out of capacity.
BADA	Bad address	The specified address is invalid: it does not appear in the list of paired DevEUI.
BADO	Bad operand	The used operandi s not supported by the command
BADV	Bad value	The value associated with the operand is invalid
DRNF	Device Reference Not found	The end-device reference given in the command line has not been found by the LWC Server
DRIV	Device Reference Invalid	The reference given by the user to the end-device is invalid
DRNA	Device Reference not Allowed	The reference given by the user is not allowed
MISP	Missing Parameters	There are some missing parameters inside the command line called by the user
DUPE	Duplicated DevEUI	The DevEui used to currently provisioned the new end-device is already in the end-device list
DUPA	Duplicated DevAddr	The DevAddr used to currently provisioned the new end-device is already in the end-device list
NOTF	Not Found	The port specified by the user is not found
PBFE	Device Packet Buffer Empty	The packet buffer for the specified end-device is empty
PBFF	Device Packet Buffer Full	The packet buffer for the specified end-device is full
FULL	Node Database full	The maximum number of device for one LWC Server is reached. The LWC Server database is full
BUSY	Dongle is busy sending	The frame to send has not been taken into account because the LWC Server was busy sending another frame

6.4 AVAILABLE RESOURCES

6.4.1 HELP (H): DISPLAYING THE LWC SERVER HELP

This command displays the available resources, or the help for a specific resource.

6.4.1.1 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{HELP|H}[<Op><Resource>]
```

Use example (applying 5.2): **GET 0 HELP**

Response:

```
<RGet><MS>0<MS><status>
```

Actions :

This command asks the display of the LWC Server's online help or a resource specific online help. The online help is displayed as several lines prefixed by one <RmkStart> character. At the end of the help, the <RmkEnd> character is displayed.

6.4.2 FIRMWARE_INFO (FWI): INFORMATION DISPLAY ABOUT THE FIRMWARE VERSION

This command asks the LWC Server to give the version of its embedded firmware.

6.4.2.1 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{FIRMWARE_INFO|FWI}
```

Use example (applying 5.2): **GET 0 FWI**

Response:

```
<RGet><MS>0<MS><status><MS>{FIRMWARE_INFO|FWI}=<Name><FS><Major><FS><Minor>
<FS><Revision>
```

Response example:

RGET 0 SUCCESS FIRMWARE_INFO=LWCServer:0.3(beta), Kernel:3.4.0.0,

FWName:lwc-server.lwcs.ClassC.lrctm.EU.chkpt.wdt2.nco.NOT_FOR_PROD-dfp-br-F5437A

6.4.3 IDENTITY (IDY): DISPLAYS THE DONGLE IDENTITY (EUI AND ADDR)

This command asks the LWC Server to display its IDs: EUI and Addr.

6.4.3.1 <GET> COMMAND

Request:

<Get><MS>0<MS>**{IDENTITY|IDY}**

Use example (applying 5.2): **GET 0 IDY**

Response:

<RGet><MS>0<MS><status><MS>**{IDENTITY|IDY}**=**<DevEUI><FS><DevAddr>**

Response example:

RGET 0 SUCCESS IDENTITY=020000FFFF007067,01007067

6.4.4 VERBOSE (VRB): LWC SERVER VERBOSITY LEVEL

This command is used to configure several LWC Server's displaying parameters.

6.4.4.1 <SET> COMMAND

Request:

```
<Set><MS>0<MS>{VERBOSE|VRB}<Op>[<Verbosity>]<FS>[<DevRefMode>]<FS>[<Echo>]
<FS>[<Debug>]
```

<Verbosity>::= {SHORT|S}|{LONG|L} : Defined the verbosity level in the commands coming from the LWC Server.

<DevRefMode>::= {DEVEUI|DE}|{DEVADR|DA}|{DEVPORT|DP} : This parameter defines how the end-devices are identified by the LWC Server during the exchanges.

<Echo>::= {ON|1}|{OFF|0} : This parameter asks to the LWC Server to copy on the serial interface all the commands that it receives.

<Debug>::= {ON|1}|{OFF|0}

<Op>::= "="

If a parameter is missing in the command (ex: SET 0 VRB=S,,1,0), then it is defined with the default parameters. The default parameters are: **LONG, DEVEUI, OFF, OFF**.

Use example: **SET 0 VRB=S,DA,1,0**

Response:

```
<RSet><MS>0<MS><status><MS>{VERBOSE|VRB}=<Verbosity><FS><DevRefMode><FS><De
bug>
```

<status>::=<StatusG> (cf. §6.3)

Response example: **RSET 0 SUCCESS VERBOSE=SHORT,DEVADDR,ON,OFF**

6.4.4.2 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{VERBOSE|VRB}
```

Use example: **GET 0 VERBOSE**

Response:

```
<RGet><MS>0<MS><status>{VERBOSE|VRB}=<Verbosity><FS><DevRefMode><FS><Echo><
FS> <Debug>
```

Response example: **RGET 0 SUCCESS VERBOSE=LONG,DEVEUI,OFF,OFF**

6.4.5 PROV_DEF_PAR (PDP): DEFAULT PROVISIONING PARAMETERS

This resource is used to configure the end-devices default provisioning parameters on the LWC Server.

6.4.5.1 <SET> COMMAND

Request:

```
<Set><MS>0<MS>{PROV_DEF_PAR|PDP}<Op><ProvDefPar>
```

```
<ProvDefPar> ::=
[<EDAM>]<FS>[<Class>]<FS>[<RX2DR>]<FS>[<NWSK>]<FS>[<APSK>]<FS>[<AppEUI>]
<FS>[<AppKey>]

<Op> ::= "="
```

If a parameter is missing in the command (ex: SET 0 PDP=OTA,C,0,,,,), then it is defined with the default parameters.

The default parameters are:

ABP, C, 0,
2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,70B3D5E75F600000,
2B7E151628AED2A6ABF7158809CF4F3C

Use example:

SET 0
PDP=OTA,A,1,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,00010203040
50607, 5B7E151628AED2A6ABF7158809CF4F4A

Response:

```
<RSet><MS><status>0
```

`<status>::=<StatusG>` (cf. §6.3)

Response example: **RSET 0 SUCCESS PROV_DEF_PAR**

6.4.5.2 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{PROV_DEF_PAR|PDP}
```

Use example: **GET 0 PDP**

Response:

```
<RGet><MS>0<MS><status><MS>{PROV_DEF_PAR|PDP}=<NwkDefPar>
```

```
<NwkDefPar> ::=
<EDAM><FS><Class><FS><RX2DR><FS><NWSK><FS><APSK><AppEUI><AppKey>
```

Response example:

RGET 0 SUCCESS
PROV_DEF_PAR=OTA,A,1,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,0
001020304050607, 5B7E151628AED2A6ABF7158809CF4F4A

6.4.6 DEV_PROV_PAR (DPP): END-DEVICE PROVISIONING MANAGEMENT

This resource is used to either get the provisioning information of an end-device or to add a new end-device to the LWC Server's provisioned end-devices list. Both OTAA and ABP are supported.

6.4.6.1 <SET> COMMAND

Request:

```
<Set><MS>0<MS>{DEV_PROV_PAR|DPP}<Op><DevParamIn>
```

```
<DevParamIn> ::=
<DevEUI><FS>[<EDAM>]<FS>[<Class>]<FS>[<RX2DR>]<FS>[<DevAdr>]<FS>[<NWSK>]<FS
>[<APSK>]<FS>[<AppEUI>]<FS>[<AppKey>]
```

It is not necessary to fill all the fields in all the cases. For example, it is not necessary to fill the AppSKey and NwkSkey field in OTAA, or it is not necessary to fill the AppEUI and AppKey field in ABP

Use example in ABP:

SET 0
DPP=70B3D5E75E000205,ABP,A,0,00000205,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6AB F7158809CF4F3C,,

Use example in OTAA:

SET 0
DPP=70B3D5E75E000205,OTA,A,0,00000205,,,70B3D5E75F600000,4B7E151628AED2A6ABF7158809CF4F5A

Response:

```
<RSet><MS>0<MS><status><MS>[{DEV_PROV_PAR|DPP}[=<DevParamOut>]
```

```
<DevParamOut> ::=
<Port><FS><DevEUI><FS><EDAM><FS><Class><FS><RX2DR><FS><DevAdr><FS><NWSK><FS
><APSK><AppEUI><FS><AppKey>
```

```
<status>::=<StatusG> |
```

SUCC: Success
NDBF: No more slot available
DUPE: Duplicate DevEUI.
DUPA: Duplicate DevAdr.

Response example in ABP:

RSET 0 SUCCESS
DEV_PROV_PAR=4,70B3D5E75E000205,ABP,A,0,00000205,2B7E151628AED2A6ABF7158809CF4F3C,2B7E15 1628AED2A6ABF7158809CF4F3C, 70B3D5E75F600000, 2B7E151628AED2A6ABF7158809CF4F3C

Response example in OTAA:

RSET 0 SUCCESS
DEV_PROV_PAR=3,70B3D5E75E0001BF,OTA,A,0,000001BF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,FFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFF,70B3D5E75F600000,4B7E151628AED2A6ABF7158809CF4F3C

Actions:

- <DevEUI> and <DevAdr> must be unique in the provisioned end-devices list
- If <DevEUI> is incorrect or not given in the command, then the <status> is BADV.
- The first free port in the LWC Server list is chosen while provisioning a new end-device
- If the command does not contains the <DevAdr> field, the latter is extracted from the <DevEUI> (it corresponds to the last 25 bits of the <DevEUI>)
- The default values for <EDAM>, <Class>, <RX2DR>, <NWSK>, <APSK>, <AppEUI> and <AppKey> are the ones that can be changed or consulted through the PROV_DEF_PAR resource (§6.4.5)

6.4.6.2 <DEL> COMMAND

Request:

<Del><MS>0<MS>**{DEV_PROV_PAR|DPP}**<Op><DevRef>

<DevRef> : Reference of the end-device to delete (DevEUI, DevAddr, Port or « All »)

Use example: **DEL 0 DPP=4**

NOTE: The use of « All » or « * » allows to delete all the provisioned end-devices list in the LWC Server **Beware, this action is non-reversible.**

Response:

<RDel><MS><status>0<MS>**{DEV_PROV_PAR|DPP}**=<DevRef>

<status>::= <StatusG> | (cf. §6.3)

NOTF: Not found.

Response example: **RDEL 0 SUCCESS DEV_PROV_PAR=4**

Actions:

- <DevRef> must exist. If it does not, the error NOTF is returned
- In SUCCESS case, the parameters and data associated to <DevRef> are completely deleted

6.4.6.3 <GET> COMMAND

Request:

<Get><MS>0<MS>**{DEV_PROV|DPP}**<Op><DevRef>

<Op> ::= "="

Use example: **GET 0 DPP=3**

Response:

<RGet><MS><status>0<MS>**{DEV_PROV|DPP}**=<DevParamOut>

```
<DevParamOut> ::=

      <Port><FS><DevEUI><FS><EDAM><FS><Class><FS><RX2DR><FS><DevAdr><FS><NW
SK><FS><APSK><FS><AppEUI><FS><AppKey>
```

Response example:

RGET 0 SUCCESS
DEV_PROV_PAR=3,70B3D5E75E000204,ABP,A,0,00000204,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,70B3D5E75F600000, 2B7E151628AED2A6ABF7158809CF4F3C

6.4.7 DEV_PROV_LIST (DPL): PROVISIONED END-DEVICES LIST MANAGEMENT

This resource allows to obtain the full list of all the provisioned end-devices on the LWC Server.

6.4.7.1 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{DEV_PROV_LIST|DPL}
```

Use example: **GET 0 DPL**

Response:

```
<RGet><MS>0<MS><status>{DEV_PROV_LIST|DPL}={<DevParam>[<LS>]}*
```

```
<DevParam> ::=

      <Port><FS><DevEUI><FS><EDAM><FS><Class><FS><RX2DR><FS><DevAdr><FS><NW
SK><FS><APSK><FS><AppEUI><FS><AppKey>
```

Response example:

RGET 0 SUCCESS DEV_PROV_LIST=

0,020000FFFF007067,ABP,C,0,01007067,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,70B3D5E75F600000,2B7E151628AED2A6ABF7158809CF4F3C;

1,70B3D5E75F0000D9,OTA,A,0,010000D9,4A551A036DF0F6CAA801656E6A7A8C37,AE68ADDEF60285D7A735ED82E14F31B7,70B3D5E75F600000,5B7E151628AED2A6ABF7158809CF4F3C;

2,70B3D5E75E0001AF,OTA,A,0,000001AF,95CA48E3C1AD49DA582F00B8A6803711,5DEA7A0971A3B455E019D845ED79DA97,70B3D5E75F600000,4B7E151628AED2A6ABF7158809CF4F3C;

3,70B3D5E75E0001BF,OTA,A,0,000001BF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,70B3D5E75F600000,4B7E151628AED2A6ABF7158809CF4F3C;

4,70B3D5E75E0000CF,ABP,A,0,000000CF,2B7E151628AED2A6ABF7158809CF4F3C,2B7E151628AED2A6ABF7158809CF4F3C,70B3D5E75F600000,2B7E151628AED2A6ABF7158809CF4F3C

6.4.8 APP: SENDING/RECEIVING APPLICATIVE FRAMES

This resource is used to send an applicative payload (or display a received applicative payload) to (or from) one of the provisioned end-devices.

The command DEL on this resource allows to empty the packet buffer stored when the destination end-device is a class A device.

6.4.8.1 <SET> COMMAND

Request:

```
<Set><MS><DevRef><MS>APP<Op><HexStr>
```

<HexStr> ::= The applicative payload presented as an hexadecimal string.

Use example: **SET 3 APP=1105001300552000**

Response:

```
<RSet><MS><DevRef><MS><status><MS>APP
```

<status>::=<StatusG> (cf. §6.3)

Response example: **RSET 00000204 SUCCESS APP**

Actions:

- If the request succeed, a frame is directly send to the end-device if the latter is a class C device. For the class A devices, the frame is stored, waiting for an uplink from the end-device.
- In case of a fail, the HOST application has to manage a buffer or a retry procedure in order to be sure that the frame has been sent.

6.4.8.2 <DEL> COMMAND

Request:

```
<Del><MS><DevRef><MS>APP
```

Use example: **DEL 3 APP**

Response:

```
<RSet><MS><DevRef><MS><status><MS>APP
```

<status>::=<StatusG> (cf. §6.3)

Response example: **RDEL 70B3D5E75F0000D9 SUCC APP**

Actions:

- When the end-device (identified by DevRef) is a class A and there are packets inside the packets buffer, then if the request succeed, all the packets for this end-device are deleted.
- If the end-device (identified by DevRef) is a class C or if there is no packets inside the packets buffer, then the LWC Server send back a “Packet Buffer Empty” error (or PBFE), cf. §6.3.

6.4.8.3 <INDIC> COMMAND

Indication:

```
<Indic><MS><DevRef><MS>APP=<Direction><FS><HexStr>
```

<Direction> : Frame direction

- {UP | U}: Uplink (End-device ➔ LWC Server)
- {DOWN | D}: Downlink (LWC Server ➔ End-device)

<HexStr> ::= The ZCL frame presented as an hexadecimal string.

Example:

INDIC 70B3D5E75E000204 APP=UP,110A00500006410903070000009B70E9204

INDIC 70B3D5E75F0000D9 APP=DOWN,110000000002

6.4.9 MAC: SENDING/RECEIVING MAC FRAMES

This resource is used to send a MAC frame (or display a received MAC frame) to (or from) one of the provisioned end-devices. This allows especially to choose the LoRaWAN port on which the frame will be sent. By default, this port is **125**.

If the port chosen by the user is **6**, then a **6lowpan header (7EF700F3450000)** is added to the frame.

The command DEL on this resource allows to empty the packet buffer stored when the destination end-device is a class A device.

6.4.9.1 <SET> COMMAND

Request:

<Set><MS><DevRef><MS>**MAC**<Op>[<FPort>]<FS><HexStr>

<HexStr> ::= The ZCL frame presented as a hexadecimal string.

Use example: **SET 3 MAC=10,1105001300552000**

Response:

<RSet><MS><DevRef><MS><status><MS>**MAC**

<status>::= <StatusG> (cf. §6.3)

Response example: **RSET 0100676A SUCCESS MAC**

Actions:

- If the request succeed, a frame is directly send to the end-device if the latter is a class C device. For the class A devices, the frame is stored, waiting for an uplink from the end-device.
- In case of a fail, the HOST application has to manage a buffer or a retry procedure in order to be sure that the frame has been sent.

6.4.9.2 <DEL> COMMAND

Request:

<Del><MS><DevRef><MS>**MAC**

Use example: **DEL 3 MAC**

Response:

<RSet><MS><DevRef><MS><status><MS>**MAC**

<status>::= <StatusG> (cf. §6.3)

Response example: **RDEL 70B3D5E75F0000D9 SUCC MAC**

Actions:

- When the end-device (identified by DevRef) is a class A and there are packets inside the packets buffer, then if the request succeed, all the packets for this end-device are deleted.
- If the end-device (identified by DevRef) is a class C or if there is no packets inside the packets buffer, then the LWC Server send back a “Packet Buffer Empty” error (or PBFE), cf. §6.3.

6.4.9.3 <INDIC> COMMAND

Indication:

```
<Indic><MS><DevRef><MS>MAC=<Direction><FS><FPort><FS><FCountUp><FS><HexStr>
```

<Direction> : Frame direction

- {UP | U}: Uplink (End-device ➔ LWC Server)
- {DOWN | D}: Downlink (LWC Server ➔ End-device)
- {JOIN REQUEST | JR}: Join Request (End-device ➔ LWC Server)
- {JOIN ACCEPT | JA}: Join Accept (LWC Server ➔ End-device)

<HexStr> ::= The ZCL frame presented as a hexadecimal string.

Example:

INDIC 01006768 MAC=UP,125,3,110A00520000410C00000000000000000000000000

INDIC 70B3D5E75F0000D9 MAC=DOWN,125,9,110000000002

6.4.10 PHY: PHYSICAL FRAME RECEPTION

This resource allows to receipt frames on a physical level. Thus, if the indication on the physical level is activated (via PHY_IND_PAR, cf. §6.4.11), the LWC Server will display every frame received at the physical level, even if the end-device sending the frame is not provisioned on the LWC Server.

6.4.10.1 <INDIC> COMMAND

Indication:

```
<Indic><MS><DevRef><MS>PHY=<Direction><FS><Freq><FS><RSSI><FS><SNR><FS><Hex Str>
```

<Direction> : Frame direction

- {UP | U}: Uplink (End-device ➔ LWC Server)
- {DOWN | D}: Downlink (LWC Server ➔ End-device)

<Freq> ::= Frequency on which the LWC Server received the frame (Hz)

<RSSI> ::= RSSI seen by the LWC Server while receiving the frame (dBm)

<SNR> ::= SNR seen by the LWC Server while receiving the frame (dBm)

<HexStr> ::= The complete physical frame received by the LWC Server without decoding the applicative payload

Example:

INDIC 000012A6 PHY=UP,868500000,-51,9,80A6120000802F007D1B91EAC10DE456BFFB2E62C48933D6

INDIC 70B3D5E75F0000D9 PHY=DOWN,868100000,,,60D90000012003007D13313648FB846335BB01

6.4.11 [APP\MAC\PHY]_IND_PAR: INDICATIONS CONFIGURATIONS FOR EACH PROTOCOL LAYER

This resource allows to select the data to display when a frame is received by the LWC Server. The resource can be developed in 3 distinct parameters for the 3 different layers used: **APP** (Application), **MAC** (Medium Access Control) or **PHY** (Physical).

For each of these layer, can be select the display, or not, of the following data:

`<Direction>`: Frame direction

- `{NONE | N}`: None
- `{UP | U}`: Uplink (End-device ➔ LWC Server)
- `{DOWN | D}`: Downlink (LWC Server ➔ End-device)
- `{BOTH | B}`: Uplink and Downlink

`<ShowPay>`: Show the payload of the received frame, or not.

- `{ON | 1}`: Show payload at corresponding layer
- `{OFF | 0}`: Do not show payload at corresponding layer

6.4.11.1 <SET> COMMAND

Request:

`<Set><MS>{0}<MS>`**`{{APP|MAC|PHY}_IND_PAR|AIP|MIP|PIP}`**`<Op>`**`<Direction>,<ShowPay>`**

- By default, only the UP frames are displayed
- For the moment, it is not possible to display DOWN frame

Use example:

SET 0 AIP=U,0
SET 0 MIP=B,0
SET 0 PIP=D,1

Response:

`<RSet><MS>{0}<MS>`**`{{APP|MAC|PHY}_IND_PAR|AIP|MIP|PIP}=<Direction>,<ShowPay>`**

`<status>::=<StatusG>` (cf. §6.3)

Response example:

RSET 0 SUCCESS APP_IND_PAR=UP,OFF
RSET 0 SUCCESS MAC_IND_PAR=BOTH,OFF
RSET 0 SUCCESS PHY_IND_PAR= DOWN,ON

6.4.11.2 <GET> COMMAND

Request:

`<Get><MS>{0}<MS>`**`{{APP|MAC|PHY}_IND_PAR|AIP|MIP|PIP}`**

Use example :

GET 0 AIP
GET 0 MIP
GET 0 PIP

Response:

`<RGet><MS>{0}<MS>`**`{{APP|MAC|PHY}_IND_PAR|AIP|MIP|PIP}=<Direction>,<ShowPay>`**

Response example:

RGET 0 SUCCESS APP_IND_PAR=UP,OFF
RGET 0 SUCCESS MAC_IND_PAR=BOTH,OFF
RGET 0 SUCCESS PHY_IND_PAR=DOWN,ON

6.4.12 LAST_RX (LRX): DISPLAY THE DELAY SINCE THE LAST RECEIVED FRAME

This resource allows to display the delay (in minutes) since the last reception of a frame from a particular end-device or for all the sensor (on the LWC Server point of view).

6.4.12.1 <SET> COMMAND

Request:

<Set><MS>{0|<DevRef>}<MS>**{LAST_RX|LRX}**

There is no operand or value. The Set Command only allows to reset the delay measure since the last reception.

Use example: **SET 2 LRX**

Response:

<RSet><MS>{0|<DevRef>}<MS><status>**{LAST_RX|LRX}**=<MinutesSinceLastUp>

<status>::= <StatusG> | (cf. §6.3)

NOTF: Not found.

<MinutesSinceLastUp> ::= Delay in **minutes** since the last frame received from an end-device in particular or from any end-device in the end-devices provisioned list. If the response does not contain a numerical value, the response strings can be:

- **NO_RX_SINCE_REINIT** : no frame received since the last initialization (reboot or SET command)
- **OVER_45_DAYS**: no frame received since at least 45 days.

Response example: **RSET 70B3D5E75E000202 SUCCESS LAST_RX=NO_RX_SINCE_REINIT**

6.4.12.2 <GET> COMMAND

Request:

<Get><MS>{0|<DevRef>}<MS>**{LAST_RX|LRX}**

Use example: **GET 1 LRX**

Response:

<RGet><MS>{0|<DevRef>}<MS><status>**{LAST_RX|LRX}**=<MinutesSinceLastUp>

<MinutesSinceLastUp> ::= Idem <Set>.

Response example: **RGET 70B3D5E75F000081 SUCCESS LAST_RX=19**

6.4.13 REBOOT (RBT): ALLOWS TO RESTART THE LWC SERVER APPLICATION

This resource allows to reboot the LWC Server application without any hardware action.

6.4.13.1 <SET> COMMAND

Request:

<Set><MS>0<MS>**{REBOOT|RBT}[**<Op>**<DelaySec>]**

<DelaySec> **::=** It is possible to specify a delay in seconds before the software reboot of the LWC Server.

Use example: **SET 0 RBT=1**

Actions:

- If Delay = 0 or empty, the reboot is done immediately
- If Delay = STOP or S, then if a reboot request is running, it is cancelled

Response:

<RSet><MS><status><MS>**{REBOOT|R}[**=**<DelaySec>]**

Response example: **RSET 0 SUCCESS REBOOT=1**

6.4.14 RX_PARAMS (RXP): ALLOWS TO MODIFY THE RX PARAMETERS

This resource allows to modify the Datarate (corresponds to a SF) used by the LWC Server, as well as the 3 listening frequencies.

6.4.14.1 <SET> COMMAND

Request:

```
<Set><MS>0<MS>{RX_PARAMS|RXP}<Op>[<Freq1>]<FS>[<Freq2>]<FS>[<Freq3>]<FS>[<R
XDR>]
```

<Freq1> ::= The frequency n°1 on which the LWC Server is listening (Hz) (from 860 MHz to 1020 MHz)

<Freq2> ::= The frequency n°2 on which the LWC Server is listening (Hz) (from 860 MHz to 1020 MHz)

<Freq3> ::= The frequency n°3 on which the LWC Server is listening (Hz) (from 860 MHz to 1020 MHz)

<RXDR> ::= The Datarate used by the LWC Server to listen *(from 0 (SF12) to 2 (SF10))*

The default values are : **RXP=868100000,868300000,868500000,0**

Use example: **SET 0 RXP=868700000,868900000,869100000,1**

Response:

```
<RSet><MS>0<MS><status><MS>{RX_PARAMS|RXP}
```

<status>::=<StatusG> (cf. §6.3)

Response example: **RSET 0 SUCCESS RX_PARAMS**

Actions:

- This command modifies the parameters used by the dongle for receiving the frames from the provisioned end-devices (frequencies and Datarate (corresponding in the LoRaWAN specifications to a Spreading Factor)).

6.4.14.2 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{RX_PARAMS|RXP}
```

Use example: **GET 0 RXP**

Response:

```
<RGet><MS>0<MS><status><MS>{RX_PARAMS|RXP}=<Freq1><FS><Freq2><FS><Freq3><FS
><RXDR>
```

Response example:

RGET 0 SUCCESS RX_PARAMS=868700000,868900000,869100000,1

6.4.15 WITH_HOST (WHT): GIVES THE HOST THE CONTROL ON THE "RE-JOIN"

This resource allows the user to decide between two modes of working for the LWC Server. Either to manage by itself when an already paired end-device tries to do another Join Request (it will send back a Join Accept). Or let the Host manages it (the LWC Server will not answer the end-device until the Host saves it again).

6.4.15.1 <SET> COMMAND

Request:

```
<Set><MS>0<MS>{WITH_HOST|WHT}<Op><State>
```

<State> ::= {ON|1}|{OFF|0} The wanted state for this option : active (ON) or unactive (OFF)

The default value is : **WITH_HOST=OFF**

Use example: **SET 0 WHT=1**

Response:

```
<RSet><MS>0<MS><status><MS>{WITH_HOST|WHT}=<State>
```

<status>::=<StatusG> (cf. §6.3)

Response example: **RSET 0 SUCCESS WITH_HOST=ON**

Actions:

- If the "With Host" option is activated, then the LoRaWAN server do not answer the JoinRequest of an already paired end-device. It notifies the Host on the serial link but do not automatically answer a Join Accept to the end-device. To accept the rejoin, the host needs to modify the provisioning parameter of this sensor the same way it adds the end-device to the LoRaWAN server the first time.
- If the "With Host" option is not activated, then the LoRaWAN server will automatically sends a Join Accept when receiving a Join Request from an already paired device.

6.4.15.2 <GET> COMMAND

Request:

```
<Get><MS>0<MS>{WITH_HOST|WHT}
```

Use example: **GET 0 WITH_HOST**

Response:

```
<RGet><MS>0<MS><status><MS>{WITH_HOST|WHT}=<State>
```

Response example:

RGET 0 SUCCESS WITH_HOST=OFF