Quick Start Guide

Getting started with ST Quuppa tag emulation

Introduction

This document provides information about the BlueNRG-LP ST Quuppa tag emulation SW provided within the STSW-QUUPPA-ETAG SW package.
The STSW-QUUPPA-ETAG evaluation SW package provides the ST Quuppa tag emulation library and associated demonstration application which allow to build a Quuppa tag emulation device supporting the following features:

- location-tracking capability
- multiple sensors data provisioning
- framework for some custom back channel commands to be used through the Quuppa's proprietary positioning system.

ST Quuppa tag emulation library is built following the specification of Quuppa Tag Emulation using Bluetooth Wireless Technology and the specification of Quuppa Tag Back Channel using Bluetooth Wireless Technology. It also allows to support a set of Quuppa preconfigured profiles which define specific tag features.

ST Quuppa tag emulation is based on Bluetooth Low Energy Advertising State as defined in Bluetooth Core Specification version 4.0 and higher. It provides support for specific standard manufacturing advertising packets (direction finding and data packets) and it works on standard Bluetooth LE channels.

This document describes the STSW-QUUPPA-ETAG SW package components and related HW and SW setup instructions.
It also provides the fundamental information about the Quuppa tag emulation and back channel capability and some details about the ST Quuppa tag emulation solution.

The following BlueNRG-LP kits are supported:

- STEVAL-IDB011V1 (QFN48 package) development platform

Contents

1 HW/SW setup

The STSW-QUUPPA-ETAG SW package is delivered as a zip file and it provides the following components:

- ST Quuppa tag emulation library folder with ST Quuppa tag emulation library, header files and demonstration application source and header files (`Projects\BLE_Examples\BLE_Quuppa_Tag_Library`).
- ST Quuppa tag emulation folder with demonstration application IDE projects IAR, KEIL and WiSE-Studio (`Projects\BLE_Examples\BLE_Quuppa_Tag_with_Lib`).
- ST Quuppa tag emulation release notes and APIs html documentation (`Docs\st_quuppa_tag_emulation_release_notes_html`, `Docs\st_quuppa_tag_emulation_apis_html`, `Docs\index_st_tag.html`).

  NOTE: `Docs\index_st_tag.html` is the entry point to ST Quuppa tag emulation html documentation.

User is requested to unzip/extract the en.STSW-QUUPPA-ETAG.zip file under his local STSW-BNRGLP-DK SW package installation folder

1.1 How to configure a ST Quuppa tag emulation?

The following instructions must be followed in order to configure a BlueNRG-LP, STEVAL-IDB011V1 kit as an ST Quuppa tag emulation device:

1) Open ST Quuppa tag emulation IDE project (IAR or KEIL or WiSE-Studio) available on `Projects/BLE_Examples/BLE_Quuppa_Tag_with_Lib` folder.
2) Power on BlueNRG-LP, STEVAL-IDB011V1 using USB (connect to a PC USB port) or by battery.
3) Build and download the ST Quuppa tag emulation on selected platform using the selected IDE download option, or just drag and drop the `BLE_Quuppa_Tag_with_Lib.hex` on the STEVAL-IDB011V1 available on Windows navigation tree, or use the RF-Flasher Utility tool.
4) If STEVAL-IDB011V1 is connected to a PC USB port, open a serial communication terminal on PC (as HyperTerminal or TeraTerm), in order to get some debug messages.
   Configuration is:
   - 115200 baud rate
   - 8 bits data
   - 1 start bit
   - 1 stop bit
   - no parity
   - no HW flow control

   NOTE: ST_QUUPPA_TAG_PROFILE_ID_BADGE is the current default Quuppa profile selected on `st_quuppa_tag_main.h` header file.

On power up, user gets a welcome message on connected hype terminal and several information messages allowing to follow device states transitions:

```
BlueNRG-LP ST Quuppa Tag Application (version: 1.0.0)
aci_gatt_srv_init() --> SUCCESS
aci_gap_init() --> SUCCESS
STORAGE state: press botton or trigger acceleration!
```

ST Quuppa tag emulation is then ready to be used within a Quuppa positioning tracking system.

2 ST Quuppa tag emulation

The ST Quuppa tag emulation demonstration example emulates a Quuppa Tag using standard Bluetooth LE specification. It configures a BlueNRG-LP ST device in order to send Quuppa Direction Finding (DF) packets used for position tracking and Quuppa Data packets which are used for data transfer.

A system employing Quuppa Intelligent Locating Technology can accurately detect and track the position of ST Quuppa tag emulation as well as receive and expose data transmitted by it.

For more information about the Quuppa emulation system and related functionalities please refer to the documentation on Quuppa web site.

NOTE:

In order to evaluate the ST Quuppa tag emulation, user could simply verify proper Direction Finding and Data packets format and contents or he could use the solution within a specific Quuppa positioning system.

The Quuppa Development Kit could be also used. In this case, user could refer to the Quuppa Development Kit User Manual for step-by-step instructions about planning, installing, deploying, and configuring a Quuppa Location System with locators and track the ST Quuppa tag emulation.

2.1 ST Quuppa tag emulation configuration

The ST Quuppa tag emulation works on standard Bluetooth LE channel 37 and it could be tracked by the Quuppa Positioning Engine (QPE) localization systems provided on Quuppa Location System.

It uses a specific public address as Quuppa tag ID which could be customized by user. The ST Quuppa developer ID is 0x0081.

A specific `ST_Quuppa_Tag_Init()` API allows to provide the following initialization parameters to the ST Quuppa tag emulation library:

- ST Quuppa tag emulation public address also used as tag ID;
- ST Quuppa tag emulation device name;
- ST Quuppa tag emulation profile number used for selecting the profile number within the supported range of profiles;
- ST Quuppa tag emulation DF packet Device Type;
- ST Quuppa tag emulation developer specific payload data used on generic developer specific payload data response;
- ST Quuppa tag emulation unsupported response payload data used on unsupported response payload.

User can customize such parameters according to the allowed values through the `st_quuppa_tag_main.h` header file.

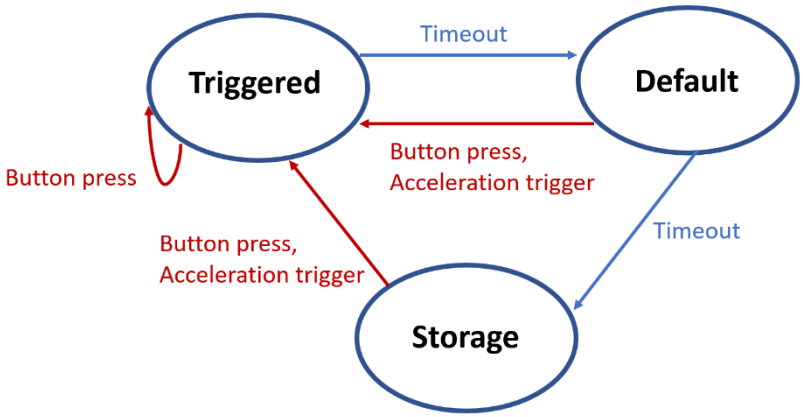
The `ST_Quuppa_Tag_Init()` API must be called at initialization phase with the selected parameters. The user must call the related ST Quuppa tag emulation state machine API `ST_Quuppa_Tag_SM()` on main `while{1}` loop.

The predefined supported profiles define the following configurations which impact how the device state machine works and some specific device RF capabilities.

The ST Quuppa tag emulation library implements the following general states defined on Quuppa Emulation Specification:

- Triggered
- Default
- Storage

Figure 1 ST Quuppa tag emulation state machine



On power up, ST Quuppa tag emulation starts according to the start state defined by the specific selected profile (refer to Section 2.2 ST Quuppa tag emulation profiles). The following states and related transitions are supported:

- Storage state with no advertising and low power mode: device kit button (PUSH1) or motion detection through device kit accelerometer allow to move to Triggered state.
- Triggered state: device has an increased direction finding advertising rate in order to improve quality on detecting position of moving tags. After a specific timeout, device moves to Default state.
- Default State: device has a reduced advertising rate which allows to track position of a stationary tag. Device kit button (PUSH1) or motion detection through device kit accelerometer allow to move to Triggered state if supported by the profile. After a specific timeout, device moves to Storage state, if this state is supported by the current profile.

NOTES:

1. Storage and Triggered states are not supported by all profiles.

2.2 ST Quuppa tag emulation profiles

The ST Quuppa tag emulation supports the following Quuppa profiles defined on Tag Emulation Pro Tag profiles specification:

1. ASSET TAG
2. ID BADGE
3. ID BADGE FAST ALARM
4. FORKLIFT/VEHICLE
5. FORKLIFT/VEHICLE FAST RX RATE

6. DEMO TAG
7. POWERSAVE

At tag initialization time, a specific profile can be selected, through the associated profile number initialization parameter, by using one of the following supported values:

- ST_QUUPPA_TAG_PROFILE_ASSET_TAG_NUM
- ST_QUUPPA_TAG_PROFILE_ID_BADGE_NUM
- ST_QUUPPA_TAG_PROFILE_ID_BADGE_FAST_ALARM_NUM
- ST_QUUPPA_TAG_PROFILE_FORKLIFT_VEHICLE_NUM
- ST_QUUPPA_TAG_PROFILE_FORKLIFT_VEHICLE_FAST_ALARM_NUM
- ST_QUUPPA_TAG_PROFILE_DEMO_TAG_NUM
- ST_QUUPPA_TAG_PROFILE_POWERSAVE_NUM

ST_QUUPPA_TAG_PROFILE_ID_BADGE_NUM is the default profile number selected on `st_quuppa_tag_main.h` file.

A profile can be also set at runtime through a specific custom Developer Specific request packet allowing to specify one of the predefined supported profiles (refer to Section 3.8 Developer Specific Set Profile Number request/response example).

The following configurations setting are used based on the selected profile, and they determine the tag behavior:

- DF packet TX rate (advertising interval)
- DF RX ON packet rate (when the specific RX_ON bit is set to 1 on DF packet)
- TX power used for transmitting packets
- Device Info packet TX rate (advertising interval) used for periodically transmitting such packets
- State transition timeout for moving from a state to another according to the device state machine transition rules.

Table 1 ASSET TAG profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	3 HZ	0.1 Hz	NA
DF RX ON packet rate	0.5 HZ	0.2 Hz	NA
TX power rate	0 dBm	0 dBm	NA
Device Info Packet TX rate	0.5 Hz	0.1 Hz	NA
State transition timeout	20 sec	NA	NA

NA: Not applicable

Table 2 ID BADGE profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	4 Hz	0.1 Hz	OFF
DF RX ON packet rate	0.5 Hz	0.1 Hz	OFF
TX power rate	0 dBm	0 dBm	OFF
Device Info Packet TX rate	0.5 Hz	0.1 Hz	OFF
State transition timeout	20 sec	2 hours	NA

NA: not applicable

Table 3 ID BADGE FAST ALARM profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	4 Hz	0.1 Hz	OFF
DF RX ON packet rate	4 Hz	0.1 Hz	OFF
TX power rate	0 dBm	0 dBm	OFF
Device Info Packet TX rate	0,5 Hz	0.1 Hz	OFF
State transition timeout	20 sec	2 hours	NA

NA: not applicable

Table 4 FORKLIFT/VEHICLE profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	9 Hz	0.1 Hz	OFF
DF RX ON packet rate	0.5 Hz	0.1 Hz	OFF
TX power rate	0 dBm	0 dBm	OFF
Device Info Packet TX rate	0.5 Hz	0.1 Hz	OFF
State transition timeout	20 sec	2 hours	NA

NA: not applicable

Table 5 FORKLIFT/VEHICLE FAST ALARM profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	9 HZ	0.1 Hz	OFF
DF RX ON packet rate	5 Hz	0.1 Hz	OFF
TX power rate	0 dBm	0 dBm	OFF
Device Info Packet TX rate	0.5 Hz	0.1 HZ	OFF
State transition timeout	20 sec	2 hours	NA

NA: not applicable

Table 6 DEMO TAG profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	9 Hz	0.2 Hz	NA
DF RX ON packet rate	5 Hz	02 Hz	NA
TX power rate	0 dBm	0 dBm	NA
Device Info Packet TX rate	0.5 Hz	0.1 Hz	NA
State transition timeout	4 sec	NA	NA

NA: not applicable

Table 7 POWERSAVE profile configuration parameters

	Triggered	Default	Storage
DF packet TX rate	NA	0.1 Hz	NA
DF RX ON packet rate	NA	0.1 Hz	NA
TX power rate	NA	0 dBm	NA
Device Info Packet TX rate	NA	0.1 Hz	NA
State transition timeout	NA	NA	NA

NA: not applicable

2.3 ST Quuppa tag emulation packets

ST Quuppa tag emulation supports the following advertising packets:

- Quuppa Direction Finding (DF) packets used for position tracking
- Quuppa Data packets which are used for data transfer.

Quuppa Direction Finding packet is a standard non-connectable Bluetooth Low Energy manufacturing advertising packet with specific information on packet payload as follow:

Figure 2 Direction Finding payload data

LSB MSB

Company ID (2 bytes)	Packet ID (1 byte)	Device Type (1 byte)	Header (1 byte)	Tag ID (1 byte)	Checksum (1 byte)	DF field (14 bytes)
0x00C7	0x01					

The header field allows to define some important parameters (TX power and TX rate) impacting the position detection and the interaction with Quuppa positioning system (Back Channel support and RX ON field)

Figure 3 Direction Finding payload header field

LSB MSB

TX Rate (2 bits)	TX Power (2 bits)	Tag ID Type (2 bits)	Back Channel Support (1bit)	RX ON (1 bit)

Quuppa Data packet is a standard non-connectable Bluetooth Low Energy manufacturing advertising packet with specific information on packet payload as follow:

Figure 4 Data packet payload

LSB MSB

Company ID (2 bytes)	Packet ID (1 byte)	Header (1 byte)	Tag ID (6 bytes)	Payload (16 Bytes)	
				Payload Type (1 byte)	Payload Data (15 bytes)
0x00C7	0xF0				

There are 2 types of Quuppa Data packet payloads: Device Info and Developer Specific.

These packets are exchanged through the Quuppa Tag Back Channel mechanism which uses the Quuppa Request (REQ) non-connectable advertising packets to carry data from Quuppa Locators to Quuppa tags and Quuppa Response (RSP) non-connectable advertising packets to carry response data from Quuppa Tags to Quuppa Locators (refer to Section 3 ST & Quuppa Back channel).

Each time the device sent a DF packet with RX_ON = 1 on header field, it scans for possible REQ packet from locator and then it replies with a RESP packet.

The following REQ/RESP packets are supported from ST Quuppa tag emulation:

1) Device Info REQ/RESP payloads which allow to request information of the ST Quuppa tag emulation
2) Back Channel Info REQ/RSP payloads which allow to request information related to the back channel of the ST Quuppa tag emulation
3) Developer Specific REQ/ RESP payloads which allow to request information of the ST Quuppa tag emulation.
    a. A generic Developer Specific REQ/ RESP framework is provided with default preconfigured date sent as developer data.
    b. A 'Set Profile Number" ST custom Developer Specific REQ/ RESP framework is provided in order to configure the specific profile number.
4) Unsupported Request RSP payload sent when Quuppa Tag receives a REQ packet that it does not support.

Device Info REQ/RESP payloads

Table 8 Device Info REQ payload data format

Byte	Field	Description
0	REQ Type	Device Info. Value=0x01
1	RFU	Value=0x00
2	RFU	Value=0x00
3	RFU	Value=0x00
4	RFU	Value=0x00
5	RFU	Value=0x00
6	RFU	Value=0x00
7	RFU	Value=0x00
8	RFU	Value=0x00
9	RFU	Value=0x00
10	RFU	Value=0x00
11	RFU	Value=0x00
12	RFU	Value=0x00
13	RFU	Value=0x00
14	RFU	Value=0x00
15	RFU	Value=0x00d

Table 9 Device Info RSP payload data format

Byte	Field	Description
0	RSP Type	Device Info. Value=0x01
1	Version	Device Info version. Value =0x00
2	Supported features	
3	Status	
4	Battery Voltage (LSB)	Battery voltage, 1 mV/bit
5	Battery Voltage (MSB)	
6	Temperature (LSB)	Temperature of the Quuppa

Byte	Field	Description
		Tag, 0.1 K/bit
7	Temperature (MSB)	
8	Developer ID (LSB)	Developer ID (0x81)
9	Developer ID (MSB)	Develop ID (0x00)
10	FW version (LSB)	minor
11	FW version (MSB)	major
12	HW version (LSB)	minor
13	HW version (MSB)	major
14	Pressure (LSB)	Air pressure, 1 Pa/bit and offset 50000 Pa
15	Pressure (MSB)	

Supported Features is used to inform about general HW features of ST Quuppa tag emulation as follow:

Table 10 Device Info RSP payload data: supported features field

Bit number	Field	Description
0	Battery Alarm ‘	0’=not supported, ‘1’ =supported
1	Button 1	0’=not supported, ‘1’ =supported
2	Button 2	0’=not supported, ‘1’ =supported
3	Batt. Voltage Meter	0’=not supported, ‘1’ =supported
4	Temperature Meter	0’=not supported, ‘1’ =supported
5	Pressure Meter	0’=not supported, ‘1’ =supported
6	RFU	0
7	RFU	0

Table 11 Device Info RSP payload data: status field

Bit number	Field	Description
0	Battery Alarm ‘	‘0’=battery OK, ‘1’ =Battery low
1	Button 1	‘0’ =not pressed, ‘1’=pressed
2	Button 2	‘0’ =not pressed, ‘1’=pressed
3	RFU	0
4	RFU	0
5	RFU	0

Bit number	Field	Description
6	RFU	0
7	RFU	0

Back Channel Info REQ/RSP payloads format

Table 12 Back Channel Info REQ payload data format

Byte	Field	Description
0	REQ Type	Back Channel Information. Value=0x02
1	RFU	Value=0x00
2	RFU	Value=0x00
3	RFU	Value=0x00
4	RFU	Value=0x00
5	RFU	Value=0x00
6	RFU	Value=0x00
7	RFU	Value=0x00
8	RFU	Value=0x00
9	RFU	Value=0x00
10	RFU	Value=0x00
11	RFU	Value=0x00
12	RFU	Value=0x00
13	RFU	Value=0x00
14	RFU	Value=0x00
15	RFU	Value=0x00d

Table 13 Back Channel Info RSP payload data format

Byte	Field	Description
0	RSP Type	Back Channel Information. Value=0x02
1	Back channel Spec number Minor	Minor
2	Back channel Spec number Minor	Major
3	Developer ID (LSB)	Developer ID (0x81)
4	Developer ID (MSB)	Develop ID (0x00)
5	RFU	Value=0x00
6	RFU	Value=0x00
7	RFU	Value=0x00
8	RFU	Value=0x00
9	RFU	Value=0x00
10	RFU	Value=0x00
11	RFU	Value=0x00

Byte	Field	Description
12	RFU	Value=0x00
13	RFU	Value=0x00
14	RFU	Value=0x00
15	RFU	Value=0x00

Unsupported Request RSP payload

Table 14 Unsupported Request RSP payload data format

Byte	Field	Description
0	RSP type	Unsupported REQ. Value=0x00
1	Back channel Spec number	Minor
2	Back channel Spec number	Major
3	Developer ID (LSB)	Developer ID (0x81)
4	Developer ID (MSB)	Developer ID (0x00)
5	DEVELOPER DATA 1	Developer specific data
6	DEVELOPER DATA 2	Developer specific data
7	DEVELOPER DATA 3	Developer specific data
8	DEVELOPER DATA 4	Developer specific data
9	DEVELOPER DATA 5	Developer specific data
10	DEVELOPER DATA6	Developer specific data
11	DEVELOPER DATA 7	Developer specific data
12	DEVELOPER DATA 8	Developer specific data
13	DEVELOPER DATA 9	Developer specific data
14	DEVELOPER DATA 10	Developer specific data
15	DEVELOPER DATA 11	Developer specific data

Developer Specific REQ/RSP payload format

Table 15 Developer Specific REQ/RSP payloads data format

Byte	Field	Description
0	REQ/RSP Payload Type	Developer Specific. Value = 0xFF
1	Developer ID (LSB)	Developer ID (0x81)
2	Developer ID (MSB)	Developer ID (0x00)
3	DEVELOPER DATA 0	Developer specific data (Default value)
4	DEVELOPER DATA 1	‘’
5	DEVELOPER DATA 2	‘’
6	DEVELOPER DATA 3	‘’
7	DEVELOPER DATA 4	‘’

Byte	Field	Description
8	DEVELOPER DATA 5	‘’
9	DEVELOPER DATA 6	‘’
10	DEVELOPER DATA 7	‘’
11	DEVELOPER DATA 8	‘’
12	DEVELOPER DATA 9	‘’
13	DEVELOPER DATA 10	‘’
14	DEVELOPER DATA 11	‘’
15	DEVELOPER DATA 12	‘’

Custom ST Quuppa tag emulation Developer Specific “Set Profile Number” REQ and RSP payloads allow to specify, at run-time, which profile has to be configured within the list of predefined supported profiles:

Table 16 Developer Specific set profile number REQ/ RESP

Byte	Field	Description
0	REQ/RSP Type	Developer Specific. Value = 0xFF
1	Developer ID (LSB)	Developer ID (0x81)
2	Developer ID (MSB)	Developer ID (0x00)
3	ST Set Profile Number REQ	0xAD (ST tag custom id for requesting a specific profile set within the supported preconfigured profiles)
4	ST Profile Number	Value in the range of supported profiles [1-7]
5	DEVELOPER DATA 2	Default value (not used)
6	DEVELOPER DATA 3	‘’
7	DEVELOPER DATA 4	‘’
8	DEVELOPER DATA 5	‘’
9	DEVELOPER DATA 6	‘’
10	DEVELOPER DATA 7	‘’
11	DEVELOPER DATA 8	‘’
12	DEVELOPER DATA 9	‘’
13	DEVELOPER DATA 10	‘’
14	DEVELOPER DATA 11	‘’
15	DEVELOPER DATA 12	‘’

3 ST & Quuppa Back Channel

Quuppa Tag Back Channel is a mechanism that allows the user of the Quuppa Intelligent Locating Technology ™ to write/read data to/from Quuppa Tags using Quuppa Web APIs available through the Quuppa Positioning Engine (QPE) server.

The Quuppa Web APIs provide a mechanism for getting several data of a project and/or Tag using HTTP requests. The requested data is returned in an easy-to-handle JSON format.

The mechanism uses then Quuppa Request (REQ) packets to carry data from Quuppa locators to Quuppa tags and Quuppa Response (RSP) packets to carry response data from Quuppa Tags to Quuppa Locators.

In particular, the Quuppa Web APIs support the Send Back Channel Request to Tag API (sendQuuppaRequest) and Retrieve Tag Replies to Back Channel Requests API (getQuuppaRequestResponse) for interacting with the specific tag.

3.1 sendQuuppaRequest Back Channel API

sendQuuppaRequest API allows to send Back Channel request to tag(s). It requires Quuppa back channel enabled.

Request packet is as follow:

/qpe/sendQuuppaRequest?tag=<tag_1,...,tag_n>&requestData=<requestBytes>[&time=timeOutMs][&humanReadable]

Request packet parameters:

`Tag`: it defines the tag id(s) to which the request is sent [Required field].

`requestData`: it defines the data payload of the request (between 1 and 16 bytes). It can be in hex format ('0x00 0x0c' or '0x000c') or in decimal format ('0 10') [Required field].

`time`: it defines how long (in ms) the system tries to command the tag (default 60000) [Optional field].

`humanReadable`: if set to true ('&humanReadable=true'), the output is formatted (indented and line-wrapped) for easier reading [Optional field].

Response is a JSON object containing the following fields in addition to base status parameters defined on Section 3.3 Base response format back channel.

Table 17 JSON object response: JSON tags array

Key	Data Type	Value
tags	JSON array	Array containing information about each of the tags to which the request was sent

Table 18 JSON object response: Tags JSON object

Key	Data type	Value
id	String	Id of the tag
Name	String	Name of the tag
sequenceNumber	Number	Running number between 0 and 15 that can be used to match the request to replies

Example

Request:

/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0x01 0xaa 0x03&humanReadable=true

Response:


```
{
"code": 0,
"command": "/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0x01 0xaa
                0x03&humanReadable=true",
"message": "Commanding 1 tag(s)",
"responseTS": 1446469174514,
"status": "Ok",
"tags": [{
   "id": "010080e18002",
   "name": null,
   "sequenceNumber": 10
}],
"version": "1.0"
}
```


3.2 getQuuppaRequestResponse Back Channel API

getQuuppaRequestResponseAPI allows to Retrieve tag replies to Back Channel requests. It requires Quuppa back channel module enabled.

Request packet is as follow:

`/qpe/getQuuppaRequestResponse?[tag=<tag_1,...,tag_n>][&humanReadable]`

Request packet parameters:

`Tag`: it defines the tag id(s) tag id(s) from which the responses are retrieved [Optional field].

`humanReadable`: if set to true ('&humanReadable=true'), the output is formatted (indented and line-wrapped) for easier reading [Optional field].

Response is a JSON object containing the following fields in addition to base status parameters defined on Section 3.3 Base response format back channel.

Table 19 JSON object response: JSON tags array

Key	Data type	Value
tags	JSON array	Array containing information about each of the tags to which the request was sent

Table 20 JSON object response: Tags JSON object

Key	Data type	Value
id	String	Id of the tag
Name	String	Name of the tag
sequenceNumber	Number	Running number between 0 and 15 that can be used to match the request to replies
replyTS	Number	Timestamp when the response was received
requestTS	Number	Timestamp when the request was sent to the tag
requestData	String	The original request payload data in hex format
replyData	String	The response payload data in hex format

Example

Request:
/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002

Response:


```
{
"code": 0,
"command":
"http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e1
  8002",
"message": "QuuppaRequests",
"responseTS": 1440409450479,
"status": "Ok",
"tags": [
{
"id": "010080e18002",
"status": "Reply Ok",
"name": "RequestTags_0002",
"quuppaRequests": [
{
"reply TS": 1440409419456,
"reply Data": "0x01",
"requestTS": 1440409418330,
"sequenceNumber": 0,
"requestData": "0x01"
},
{
"reply TS": 1440409432683,
"reply Data": "0x01",
"requestTS": 1440409432094,

"sequenceNumber": 1,
"requestData": "0x01"
},
{
"reply TS": 1440409447347,
"reply Data": "0x05",
"requestTS": 1440409446236,
"sequenceNumber": 2,
"requestData": "0x05"
},
```

null,
null,
null,
null,
null,
null,
null,
null,
null,
null,
null,
null,
null
]
}
],
"version": "1.0"
}

3.3 Base status parameters response format

Base status parameters responses are defined as on following table:

Table 21 Base status parameters: response format

Param name	Param format	Param value
code	number	Status code, 0 if command successful, otherwise refer to Table 15 Base response parameters: status field
status	string	Status string. Refer to Table 15 Base response parameters: status field
message	string	Custom status message trying to explain error situations
version	string	version string Version of this response, in form of major, minor. Higher major versions are NOT backwards compatible to low er major versions, but higher minor
command	string	Command that was used to create this result
responseTS	number	Timestamp when this response was generated.

Table 22 Base status parameters: status code

Code	Status string
0	Ok
1	General Failure
2	Not Implemented
3	License Does Not Allow
4	Invalid Parameter
5	Required Parameter Missing

Code	Status string
6	Too Many Parameters
7	Object Not Found
8	Group Empty
9	Already Commanded
10	IO Failure
11	PENotStarted
12	Not Supported
13	Unknow nTag Listed
14	Unknow nLocator Listed
15 onwards	RFU

Example


```
{
  "code": 0,
  "command": "http://localhost:8080/qpe/commandTag?tag=010080e18002&id=superMarketActiv
   ation",
  "message": "Commanding 1 tag(s)",
  "responseTS": 1430142139012,
  "status": "Ok",
  "version": "1.0"
}
```


3.4 Device Info request/response example

Request for Device Info (request type value: 0x01)


```
{
  "code": 0,
  "command":
 "http://localhost:8080/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0x01&human
   Readable=true",
  "message": "Commanding 1 tag(s)",
  "responseTS": 1642753883483,
  "status": "Ok",
  "tags": [{
    "sequenceNumber": 0,
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002"
  }],
  "version": "1.0”
}
```


Response for Device Info (request type value: 0x01)


```
{
  "code": 0,
  "command":
  "http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002",
```

```
  "message": "QuuppaRequests",
  "responseTS": 1642753889840,
  "status": "Ok",
  "tags": [{
    "quuppaRequests": [
      {
        "sequenceNumber": 0,
        "requestTS": 1642753883483,
        "replyTS": 1642753888317,
        "replyData": "0x01 0x01 0x32 0x00 0x00 0x00 0xae 0x0b 0x81 0x00 0x00 0x01 0x00 0x01
                      0xb4 0xcb",
        "requestData": "0x01"
      },
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ],
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002",
    "status": "ReplyOk"
  }],
  "version": "1.0"
}
```


3.5 Back Channel Info request/response example

Request for Back Channel (request type value: 0x02)


```
{
  "code": 0,
  "command":
"http://localhost:8080/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0x02&human
   Readable=true",
  "message": "Commanding 1 tag(s)",
```

```
  "responseTS": 1642753923849,
  "status": "Ok",
  "tags": [{
    "sequenceNumber": 1,
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002"
  }],
  "version": "1.0"
}
```


Response for Back Channel (request type value: 0x02)


```
{
  "code": 0,
  "command":
  "http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002",
  "message": "QuuppaRequests",
  "responseTS": 1642753930684,
  "status": "Ok",
  "tags": [{
    "quuppaRequests": [
      ….,
      {
        "sequenceNumber": 1,
        "requestTS": 1642753923849,
        "replyTS": 1642753924339,
        "replyData": "0x02 0x01 0x01 0x81 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
                      0x00 0x00",
        "requestData": "0x02"
      },
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ],
    "name": "ST_Tag_BLE_New_version",
```

```
    "id": "010080e18002",
    "status": "ReplyOk"
  }],
  "version": "1.0"
}
```


3.6 Unsupported request/response example

Request for Unsupported (request type value: 0xAA as example)


```
{
  "code": 0,
  "command":
"http://localhost:8080/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0xAA%200x81%200x00&humanReadable=true",
  "message": "Commanding 1 tag(s)",
  "responseTS": 1642754342912,
  "status": "Ok",
  "tags": [{
    "sequenceNumber": 5,
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002"
  }],
  "version": "1.0"
}
```


Response for Unsupported (request type value: 0xAA as example)


```
{
  "code": 0,
  "command":
  "http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002",
  "message": "QuuppaRequests",
  "responseTS": 1642754385215,
  "status": "Ok",
  "tags": [{
    "quuppaRequests": [
      ….,
      {
        "sequenceNumber": 5,
        "requestTS": 1642754342911,
        "replyTS": 1642754349333,
        "replyData": "0x00 0x01 0x01 0x81 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
                      0x0a 0x0b",
        "requestData": "0xaa 0x81 0x00"
      },
      null,
      null,
```

```
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ],
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002",
    "status": "ReplyOk"
  }],
  "version": "1.0"
}
```


3.7 Developer Specific request/response example

Request for Developer Data (request type value: 0xFF)


```
{
  "code": 0,
  "command":
"http://localhost:8080/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0xFF%200x81%200x00&humanReadable=true",
  "message": "Commanding 1 tag(s)",
  "responseTS": 1642753992164,
  "status": "Ok",
  "tags": [{
    "sequenceNumber": 2,
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002"
  }],
  "version": "1.0"
}
```


Response for Developer Data (request type value: 0xFF)


```
{
  "code": 0,
  "command":
  "http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002",
  "message": "QuuppaRequests",
  "responseTS": 1642754000072,
  "status": "Ok",
  "tags": [{
    "quuppaRequests": [
      …,
```

```
      {
        "sequenceNumber": 2,
        "requestTS": 1642753992164,
        "replyTS": 1642753997612,
        "replyData": "0xff 0x81 0x00 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a
                      0x0b 0x0c",
        "requestData": "0xff 0x81 0x00"
      },
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ],
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002",
    "status": "ReplyOk"
  }],
  "version": "1.0"
}
```


3.8 Developer Specific Set Profile Number request/response example

Request for Developer Data with REQUEST (0xAD) for Profile Number setting to ID BADGE (0x02)


```
{
  "code": 0,
  "command":
"http://localhost:8080/qpe/sendQuuppaRequest?tag=010080e18002&requestData=0xFF%200x81%200x00%200xAD%200x02&humanReadable=true",
  "message": "Commanding 1 tag(s)",
  "responseTS": 1642754368939,
  "status": "Ok",
  "tags": [{
    "sequenceNumber": 4,
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002"
  }],
```

```
  "version": "1.0"
}
```


Response for Developer Data with REQUEST (0xAD) for Profile Number setting to ID BADGE (0x02)


```
{
  "code": 0,
  "command":
  "http://localhost:8080/qpe/getQuuppaRequestResponse?humanReadable&tag=010080e18002",
  "message": "QuuppaRequests",
  "responseTS": 1642754385215,
  "status": "Ok",
  "tags": [{
    "quuppaRequests": [
      …,
      {
       "sequenceNumber": 4,
       "requestTS": 1642754213330,
       "replyTS": 1642754222405,
       "replyData": "0xff 0x81 0x00 0x00 0x01 0x02 0xad 0x02 0x05 0x06 0x07 0x08 0x09 0x0a
                     0x0b 0x0c",
       "requestData": "0xff 0x81 0x00 0xad 0x02"
      },
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ],
    "name": "ST_Tag_BLE_New_version",
    "id": "010080e18002",
    "status": "ReplyOk"
  }],
  "version": "1.0"
}
```

4 Acronyms and abbreviations

Table 23 List of acronyms

Acronym	Description
Bluetooth LE	Bluetooth Low Energy
DF	Direction Finding
DK	Development kit
QPE	Quuppa Positioning Engine
REQ	Request
RSP	Response
SW	Software
USB	Universal serial bus

5 References

Table 24 References

What	Where	Description
STSW-QUUPPA-ETAG	www.st.com/content/st_com/en/products/embedded-software/evaluation-tool-software/stsw-quuppa-etag.html	ST Quuppa tag emulation SW web page
BlueNRG-LP Bluetooth Low Energy wireless System on Chip	www.st.com/bluenrg-lp	BlueNRG-LP device web page
STEVAL-IDB011V1	www.st.com/bluenrg-lp, Tools and Software, Solution Evaluation Tools section	STEVAL-IDB011V1 platform web page
STSW-BNRGLP-DK	www.st.com/bluenrg-lp, Tools and Software, Evaluation Tool Software section	BlueNRG-LP DK SW package web page
UM2735	www.st.com/resource/en/user_manual/dm00711446-bluenrglp-development-kits-stmicroelectronics.pdf	BlueNRG-LP Development Kits user manual
Quuppa Tag Emulation	Quuppa customer portal	Specification of Quuppa Tag Emulation using Bluetooth Wireless Technology
Quuppa Tag Back Channel	Quuppa customer portal	Specification of Quuppa Tag Back Channel using Bluetooth Wireless Technology
Quuppa Web APIs	Quuppa customer portal	APIs available through the Quuppa Positioning Engine (QPE) server.
Quuppa Tag Emulation Pro profiles	Quuppa customer portal	Specification of Quuppa Profiles specification
Quuppa Development Kit Quick Start Guide	www.quuppa.com	Quick start guide about how to setup a Quuppa Positioning tracking system with Quuppa development kit HW and SW components

6 Revision history

Table 25 Revision history

Date	Revision	Change
10-February-2022	1	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.