# ThingPark Wireless
# AS Tunnel Mode API

Revision: 1.1

Date: 2015/07/22

## Versioning

| Version | Date | By | |
|---|---|---|---|
| **1** | 09/07/2015 | SMU | Extracted Tunnel Mode Interface from ThingPark Wireless Application Developer Guide (rev 11) |
| **1.1** | 22/07/2015 | SMU | Added sample curl commands to turn ON/OFF the mote LED<br>Added section 1.2.2 describing the queueing of several messages |

# Contents

1 The ThingPark Wireless tunnel mode interface ..... 3
1.1 LRC frame tunneling interface ..... 3
1.1.1 Uplink interface: LRC to tunnel mode application server ..... 3
1.2 Downlink interface ..... 5
1.2.1 Downlink confirmed Application server payload ..... 6
1.2.2 Downlink multicast ..... 6

# 1 The ThingPark Wireless tunnel mode interface

Two level of interfaces are provided by the ThingPark® Wireless platform for developers of applications interfaced with wireless sensors or actuators compatible with the LoRaWAN specification:

- **Tunnel mode interface** : A simple message passing interface between the ThingPark Wireless servers which implement the network MAC layer (LRC servers), and application servers. This interface forwards the uplink radio packets raw payload data and associated metadata (RSSI, SNR, …) to one or more application servers associated to the network node MAC address. As ThingPark® Wireless supports bidirectional communications, applications servers may also send requests to one of the LRC nodes to send downlink frames to a network node identified by its full format (64bits) MAC address. This interface may be used when a device is associated to a single type of IoT cloud application, and therefore the payload format may be arbitrary.
- **ETSI M2M open data REST interface**: This interface must be used to leverage the additional M2M cloud services provided by the ETSI M2M platform, such as big data historization, Subscribe/Notify, sharing of sensors by multiple application servers, and secure open data APIs to third party consumers.

This document focuses on the tunnel mode interface. It provides information on:

- Low level LRC configuration parameters required to associate one or more application servers with a given network node MAC address.
- Format of LRC to application server messages that encapsulate uplink payload data and associated metadata
- The format of application server to LRC messages that encapsulate downlink payload.

## 1.1 LRC frame tunneling interface

The tunneling interface is based on HTTP for uplink & downlink packets flows. On the production platform, an IP VPN is setup on a case by case basis to secure the connection between the application server and the LRC cluster, contact Actility operations to get this VPN configured for your application server.

### 1.1.1 Uplink interface: LRC to tunnel mode application server

Uplink destination URIs are defined for each device in service profiles (basically one per device). The main criteria is the LoRa port numbers expressed as intervals, lists or single values. A default LoRa port destination can be declared.

There are two modes of distribution which are not exclusive:

- Blast/blind mode: packets are delivered to a set of destinations without acknowledgment control.
- Order mode: packets are delivered to a list of destinations until one of them confirms receipt (200OK).

Duplicate packets (same counter up and same payload) are not sent to application servers, as long as the multiple copies are received by the network infrastructure within a maximum delay of 250ms[1]

[1] In case of extreme jitter (e.g. base stations connected to the cloud via cellular network backhaul), it is possible that the network will send duplicate copies of the same frame to the application server.

(configurable by the LPWA operator). The XML payload sent to the application servers still include the RF metadata corresponding to all receiving base stations.

The uplink message is transmitted in a HTTP/POST request with query parameters and an XML payload.

**Query parameters:**

```
LrnDevEui: 00000000007E074F        // the device DevEUI
LrnFPort: 2                        // the LoRaWAN port number
LrnInfos: UPHTTP_LAB_LORA          // the service profile name used to route the packet
```

**XML payload**

```
<?xml version="1.0" encoding="UTF-8"?>
<DevEUI_uplink xmlns="http://uri.actility.com/lora">
  <Time>2015-07-09T16:06:38.49+02:00</Time>          // timestamp for the packet
  <DevEUI>00000000007E074F</DevEUI>
  <FPort>2</FPort>                                   //LoRaWAN port number
  <FCntUp>11</FCntUp>                                // the uplink counter for this packet
  <ADRbit>1</ADRbit>
  <FCntDn>0</FCntDn>                                 // the last downlink counter to the device
  <payload_hex>0027…bd00</payload_hex>                //LoRaWAN payload in hexa ascii format
  <mic_hex>38e7a3b9</mic_hex>                        // MIC in hexa ascii format
  <Lrcid>00000065</Lrcid>
  <LrrRSSI>-60.000000</LrrRSSI>
  <LrrSNR>9.750000</LrrSNR>
  <SpFact>7</SpFact>
  <SubBand>G1</SubBand>
  <Channel>LC2</Channel>
  <DevLrrCnt>3</DevLrrCnt>                           // number of LRRs which received this packet
  <Lrrid>08040059</Lrrid>
  <LrrLAT>48.874931</LrrLAT>
  <LrrLON>2.333673</LrrLON>
  <Lrrs>
    <Lrr>
      <Lrrid>08040059</Lrrid>
      <LrrRSSI>-60.000000</LrrRSSI>
      <LrrSNR>9.750000</LrrSNR>
    </Lrr>
    <Lrr>
      <Lrrid>33d13a41</Lrrid>
      <LrrRSSI>-73.000000</LrrRSSI>
      <LrrSNR>9.750000</LrrSNR>
    </Lrr>
    <Lrr>
      <Lrrid>a74e48b4</Lrrid>
      <LrrRSSI>-38.000000</LrrRSSI>
      <LrrSNR>9.250000</LrrSNR>
    </Lrr>
  </Lrrs>
  <CustomerID>100000507</CustomerID>
  <CustomerData>…</CustomerData>                     // ascii customer data set by provisioning
```

```
  <ModelCfg>0</ModelCfg>
</DevEUI_uplink>
```

## 1.2 Downlink interface

Depending on the device provisioning (application secret keys) encryption/decryption can be performed by the LRC.

The following HTTP/POST message format is used to tunnel the radio frame payload and associated metadata from the target application server to the LRC. The downlink destination URI is the primary LRC cluster (lrc1.thingpark.com) or the backup LRC (lrc2.thingpark.com) in case of failure of the primary LRC cluster. The application server acts as a HTTP client and the LRC acts as a HTTP server.

The HTTP POST format corresponds to the v1.0 production LRC platform. The LoRaWAN MAC message integrity code (MIC) is always computed by the LRC, as part of the MAC frame formatting. The MAC payload may be encrypted either by the application (default value : end to end encryption), or by the LRC if an AppKey is configured for the desired application port FPort.

Such POST command may be generated easily by tools such as curl or POSTman.

```
curl -H "Content-type:application/x-www-form-urlencoded" -X POST
http://lrc1.thingpark.com:8807/sensor/?DevEUI=000000000F1D8693&FPo
rt=1&Payload=0102030405060708090A0B0C0D0E0F&FCntDn=1234
```

Query parameters:

- DevEUI (Mandatory): target device IEEE EUI64 in hexadecimal form (representing 8 octets)
- FPort (Mandatory): target port (in decimal format)
- Payload (Mandatory): hexadecimal payload. The hexadecimal payload will be encrypted by the LRC cluster if FCntDn parameter is absent, and if the LRC has been configured with an AppSKey for the specified LoRaWAN port, otherwise the Payload must be encrypted by the Application Server according to the LoRaWAN specification. The Application Server encryption uses the downlink counter, which is why the FCntDn query parameter is required in this case.
- FCntDn (Optional); LoRaWAN Downlink Counter value used to encrypt the payload. This query parameter is needed only if the Application server (not the LRC) encrypts the payload. If present, FCntDn will be copied in the LoRaWAN header field FCnt, and the encrypted payload will be copied as-is to the LoRaWAN downlink frame by the LRC.
- Confirmed (Optional). A value of Confirmed=0 requests transmission of an UNCONFIRMED downlink frame. A value of Confirmed=1 requests transmission of a CONFIRMED downlink frame. Default value Confirmed=0 (UNCONFIRMED).
  Support of Confirmed frame transmission is subject to Connectivity plan feature flag "ackedDownlinkFrame": if the Confirmed flag is set on the HTTP POST and the device is associated with a Connectivity plan where the "ackedDownlinkFrame" feature flag is set, the downlink packet is processed, otherwise the processing is aborted and a specific error code is returned to the AS in the HTTP response.

LRC HTTP response codes:

- 200 “Request queued by LRC”: request accepted and queued until the class A device opens Rx slots by sending an uplink. In the case of a class C device, the downlink command will be

sent as soon as the LRR base station radio is available and the maximum regulatory Tx duty cycle allows transmission.

- 350 “Invalid DevEUI”
- 350 “Downlink counter value already used. Expected=1238”: the downlink counter value was already used, for instance due to a race condition with another Application server.
- 350 “Downlink counter value increment too large. Expected=1001”: the AS supplied downlink counter value is much larger than the expected downlink counter value and was rejected by the LRC.
- 350 "Confirmed downlink is not authorized for this device" : the request for transmission of a confirmed downlink packet was rejected by the LRC due to absence of “ackedDownlinkFrame” feature flag in the Connectivity plan associated to the device.

Sample CURL Command to turn ON the yellow LED on the mote with devEUI 000000000D177804

```
curl -H "Content-type:application/x-www-form-urlencoded" -X POST
http://lrc1.thingpark.com:8807/sensor/?DevEUI=000000000D177804&FPort=1&Payload=01
```

Use the same command with Payload=00 to turn off the LED.Note regarding queueing of several messages:The ThingPark Wireless network may queue up to 5 messages per device. The network uses the FPending flag defined in the LoRa WAN protocol to signal to the device that additional messages are queued. Messages will be sent, one at a time, in the receive window following the next uplink message received from the device.

### 1.2.1 Downlink confirmed Application server payload

Unconfirmed Downlink messages are not acknowledged at LoRaWAN level and therefore the network, and the tunnel mode Application server does not know whether they have been received or not.

Confirmed Downlink messages are acknowledged by the target device, but LoRaWAN section 4.3.1.2 “Message acknowledge bit and acknowledgement procedure (ACK in FCtrl)” lets the device free of sending delayed ACKs. Therefore it is not possible to let the network manage retransmissions.

When the LRC receives a possibly empty (no payload) uplink message with ACK set in the FCtrl field, the LRC will add a “ACKbit” flag in the XML metadata of the packet sent to the Application server. The retransmit policy is up to the application server.

### 1.2.2 Downlink multicast

At Phy level Multicast support in LoRaWAN class C amounts to having multiple End devices listen to the same network address. This is already supported as part of class C support in ThingPark Wireless, an Application server just needs to send an unconfirmed downlink message to the target group address (configured as dummy device).

However the LoRaWAN roadmap includes future work on multicast, including group membership and key management procedures, as well as large payload fragmentation transmission e.g. for firmware updates. Actility is an active contributor to this work and will implement in ThingPark Wireless once the LoRaWAN multicast specification is published.