ThingPark Wireless Evolved Packet Core (EPC) Connector Product Description

Under NDA

NOTICE

Headquarters

Actility Lannion,

Actility S.A 4 rue Ampère BP 30225

22300 Lannion France

www.actility.com

VERSIONS

Version	Date	Author	Details
01	31/03/17	Rohit Gupta	Initial Version

REFERENCE DOCUMENTS

	Documents	Author
01	ThingPark Product Description	Shmuel Solomon
02	ThingPark Product Version Release Notes	Laurent Barbot
03	ThingPark Wireless Device Manager User Guide	Shmuel Solomon
04	HSS/SPR function Introduction	Bastien Mercier
05	HSS/SPR Admin Guide	Bastien Mercier
06	HSS/SPR Installation Manual	Bastien Mercier
07	PGW function Introduction	Bastien Mercier
08	PGW Admin Guide	Bastien Mercier
09	PGW Installation Manual	Bastien Mercier
10	LRC AS Tunnel API	Gilles Lefevre
11	ThingPark OSS/BSS API Specification	Shmuel Solomon
12	How to build a multi-technology Scalable IoT connectivity Platform?	Rohit Gupta

WHAT'S NEW

New/Enhanced Functionalities	For More Information, See…	Release

TABLE OF CONTENTS

Acronyms and Definitions

Acronyms	Definitions
ARPU	Average Revenue Per User
AS	Application Server
BSS	Billing Support Systems
CSP	Communication Service Provider
EPC	Evolved Packet Core
eSIM	Electronic Subscriber Identity Module
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
ISM	Industrial Scientific Medical
GSCL	Gateway Service Capability Layer
LPWAN	Low Power Wide Area Network
LRC	Long Range Controller
M2M	Machine-2-Machine
MAC	Media Access Control
MCC	Mobile Country Code
MNC	Mobile Network Code
NIDD	Non-IP Data Delivery
NW	Network
OCS	Online Charging System
OFCS	Offline Charging System
OSS	Operations Support Systems
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
PGW	Packet Gateway
PKI	Public Key Infrastructure

Acronyms	Definitions
POC	Proof Of Concept
REST	Representational State Transfer
SCEF	Service Capability Exposure Function
SLRC	Secured LRC (VPN Concentrator)
SMP	System Management Platform
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNR	Signal to Noise Ratio
SPR	Subscriber Profile Repository
TWA	ThingPark Wireless Application
UICC	Universal Integrated Circuit Card
USIM	Universal Subscriber Identity Module
VPN	Virtual Private Network

1 Scope

This document describes the ThingPark Wireless Evolved Packet Core (EPC) Connector solution which is a module within ThingPark Wireless to interface with 3GPP core network, and addresses the following topics:

- Architecture Description
- Solution Configuration
- Feature limitation and expected roadmap enhancements

2 THINGPARK WIRELESS: A CONVERGED SERVICE & DATA MANAGEMENT FRAMEWORK FOR LPWAN CONNECTIVITY

ThingPark Wireless offers a converged IoT platform seamlessly integrating LoRaWAN™ and Cellular IoT technologies. ThingPark Wireless presents a unified user interface and APIs to applications, and single layer of device and connectivity management for either both LoRaWAN™ and cellular technologies. It exhibits the following high-level features:

- Converged Platform agnostic to radio to seamlessly manage both LoRaWAN™ and Cellular IoT technologies
- OSS/BSS Solution with focus on IoT
- Data Mediation layer for building data analytics and interfacing with 3rd party cloud servers (for ex. Amazon AWS)
- Pre-integrated interface with ThingPark Market enables acceleration of operator goto market through dynamic open ecosystem, and taping into the whole service value, not just connectivity
- Billing solution focused towards IoT
- Open and modular with OSS/BSS APIs allowing easy integration with operator's internal or 3rd party platforms/applications

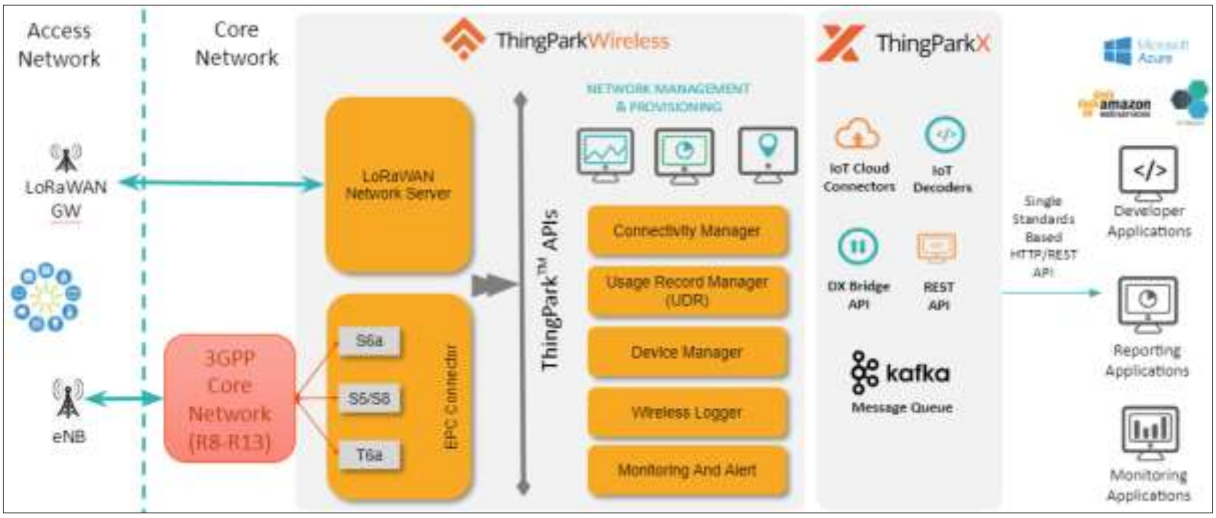


Figure 1 : ThingPark Wireless Architecture

ThingPark Wireless enables Operators to build value chain beyond connectivity

2.1 EPC Connector: Overview

EPC Connector is an add-on within ThingPark Wireless that provides insertion into existing 3GPP mobile operator core using standard-compliant interfaces (S6a, S5/S8, T6a) and provides unified device and connectivity management for IoT devices.

ThingPark wireless integrates both the EPC Connector and LoRaWAN™ network server and provides a unified interface to IoT subscribers which have both LoRaWAN™ and 3GPP (NB-IoT/Cat-M1) based deployments.

ThingPark wireless integrates seamlessly with ThingPark X, allowing a unified data management framework with connectivity towards apps and cloud platforms (for ex. Microsoft Azure, IBM Bluemix and Amazon AWS).

The key benefits are:

- Single layer of device provisioning
- Unique connectivity management layer related services
- Homogeneous interface to Application layers
- Unique & Consistent LPWAN Usage Record and Policy management layer
- Billing/rating capabilities specifically tailored for IoT
- Unique cost-effective licensing model to replace traditional HSS/PGW/PCRF/PCEF modules within 3GPP Core network which are too expensive for IoT use cases
- OSS/BSS APIs that can be used to pre-integrate 3rd party or legacy operator platforms

2.2 EPC Connector: Multi-tenancy

Typically, service providers sell IoT connectivity to enterprise customers. This requires a multi-tenant platform that enables sharing of the same infrastructure for multiple enterprise customers.

The following graphic shows the multi-tenancy concept in ThingPark wireless which can be used to host multiple enterprise clients within the same platform. Multi-tenancy is the key to providing horizontal IoT platform that can serve different IoT verticals, each representing markets with diverse requirements.

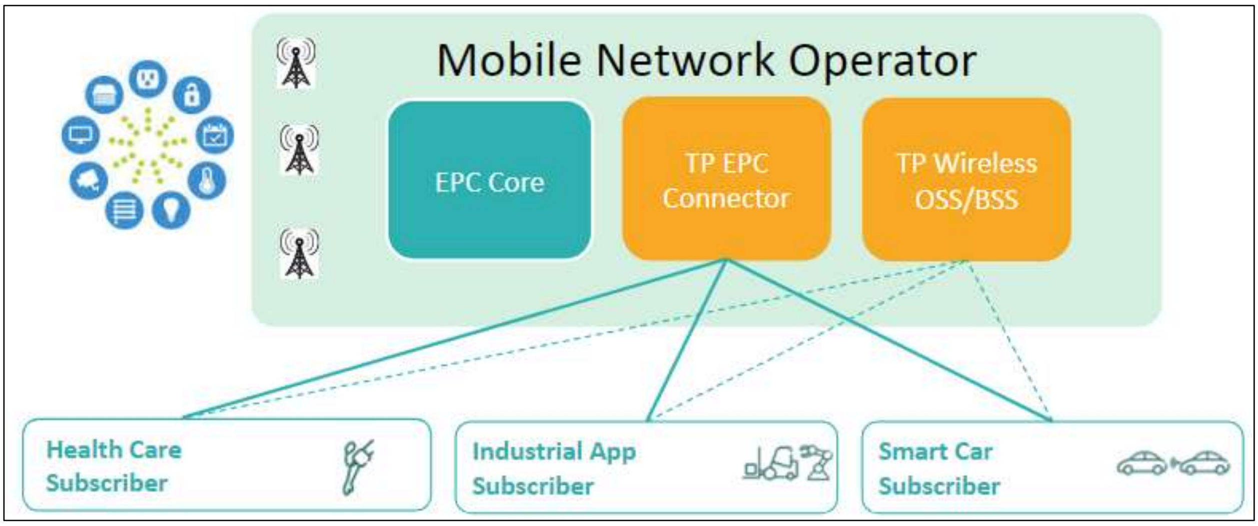


Figure 2 : Multi-tenancy support

Multi-tenancy is the key for operator scaling its B2B business with Enterprise customers

2.3 EPC Connector: Modes of Communication

LoRaWAN™ is a non-IP protocol purely based on exchanging small messages between LoRaWAN™ devices and application servers.

Cellular IoT is IP-based, but new proposals such as Non-IP Data Delivery (NIDD) in R13 3GPP standards enable efficient use in message mode. In order to optimize any Cellular IoT communication pattern, EPC Connection supports two modes of communication:

- **Message mode:** In this mode, devices exchange small messages (NIDD) or based on UDP payload with the application server.
- **Direct IP mode:** In this mode, EPC Connector routes the traffic directly to Application Server Router (ASR) via Internet, which routes it further to Application servers sitting behind firewall for security reasons. However, it sends charging information in real-time to ThingPark wireless applications.

The modes of communication for different devices/IoT subscribers are configured within ThingPark Wireless using the routing profile and connectivity plan objects, which are also used for LoRaWAN™. This unifies data records, traffic enforcement, traffic management seamlessly between LoRaWAN™ and Cellular devices to give a consistent experience for device and connectivity management to IoT subscribers.

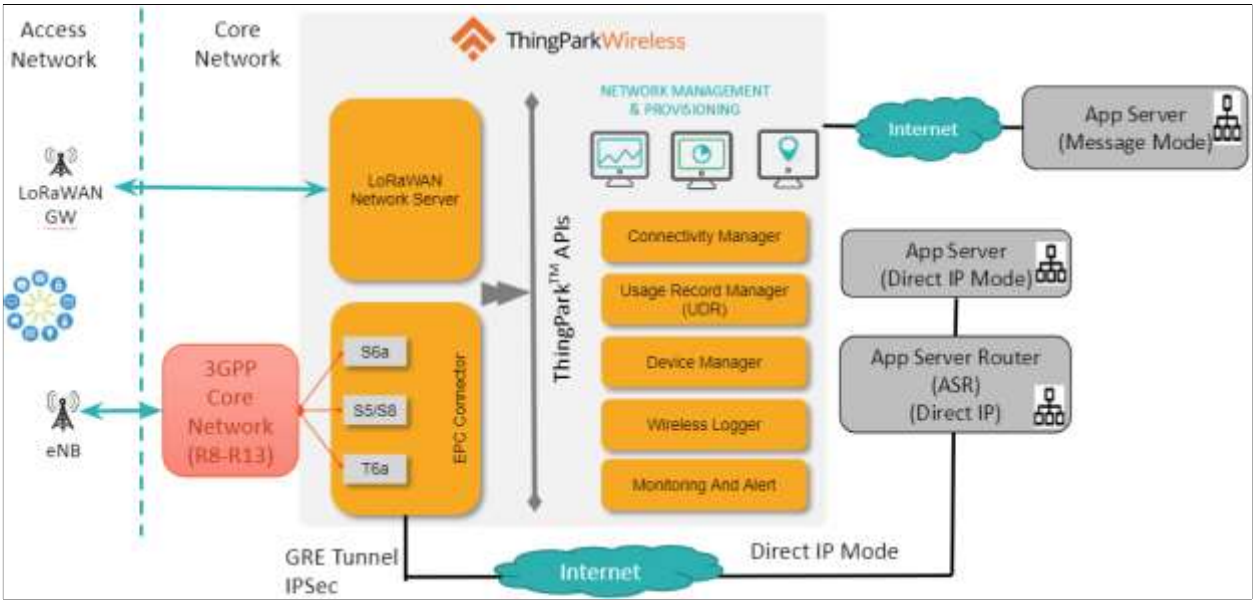


Figure 3 : Modes of Communication

2.4 EPC Connector: Key Enabler for billing features for Cellular IoT

Traditional billing systems within operator networks are designed for human-centric communications and do not meet the needs of IoT applications. However, upgrading the traditional billing systems is a costly exercise which can jeopardize the ROI on IoT investments for an operator. Actility has developed an optimized billing system closely integrated with ThingPark wireless allowing operators to monetize IoT connectivity easily from the ThingPark platform with minimal investment compared to traditional billing system upgrades, and with immediate time to market.

In order to adapt to the diversity of connectivity use cases, the EPC connector and ThingPark platform provide two options for usage records and policy enforcement:

- **Message mode:** this is the optimal mode for non-IP Cellular and LoRaWAN™. Accounting is based on message count, and policy enforcement reports number of messages that did not comply to token bucket regulator, which therefore can be charged at a higher rate.
- **Direct IP mode:** this is the optimal mode for applications using traditional IP connectivity.

To provide accounting records for the flow mode (“Direct IP” mode), the EPC connector implements the concept of “microflow” to report the traffic statistics on a message queue. Each “Microflow” record characterizes the real-time traffic within a certain time period and may be generated also based on volume triggers.

The charging metrics for Direct IP mode and message mode are shown in Figure 4. Charging metrics are generated per device and also per IoT subscriber.

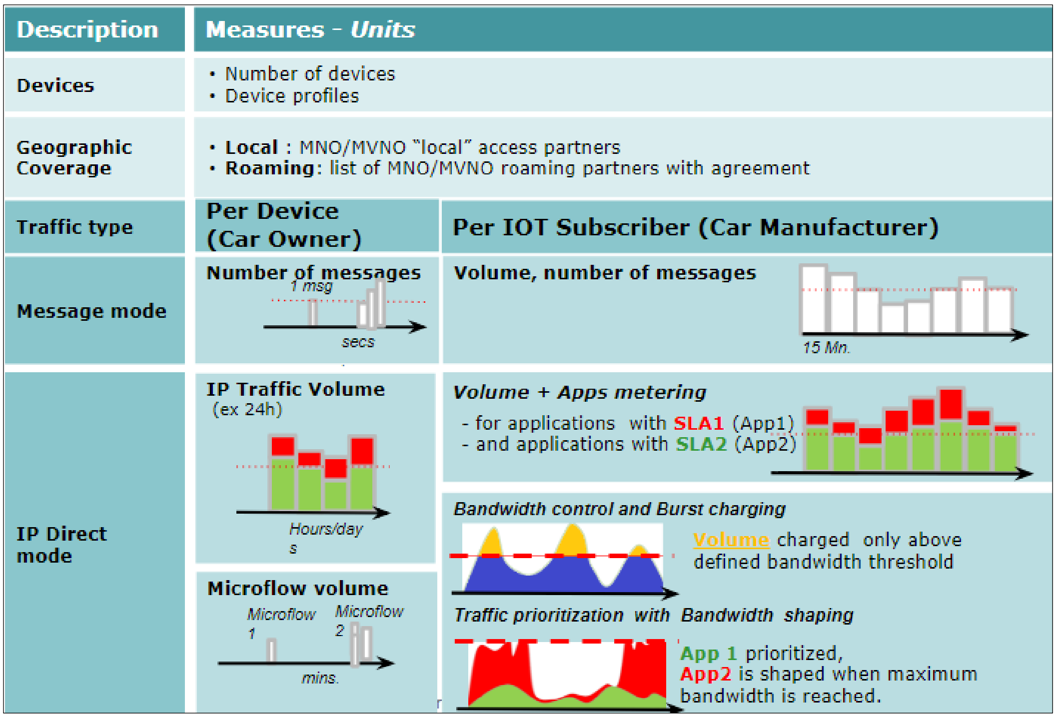


Figure 4 : Charging Criteria

For the case of "Direct IP" mode, charging metrics can be generated per period, or based on real-time microflow reports.

A given Subscriber may own devices that require different SLAs and priority levels. The EPC connector subscriber level charging records report aggregate usage per SLA level and allow differentiated aggregate charging.

The EPC connector also supports diverse business models for the Subscriber aggregate bandwidth:

- Some subscribers prefer deterministic bills and require a fixed cap to the bandwidth: the EPC connector policy enforcement module limits the aggregate bandwidth, but still provides better service to devices associated to prioritized SLAs.
- Some subscribers have bursty aggregate traffic: the EPC connector enables flexible business models where the Subscriber can be charged at a lower flat rate below a certain aggregate bandwidth usage, but pay a variable charge for traffic volume above the bandwidth limit. This model may be used to take into account periodic bursts (for ex. due to firmware upgrades).

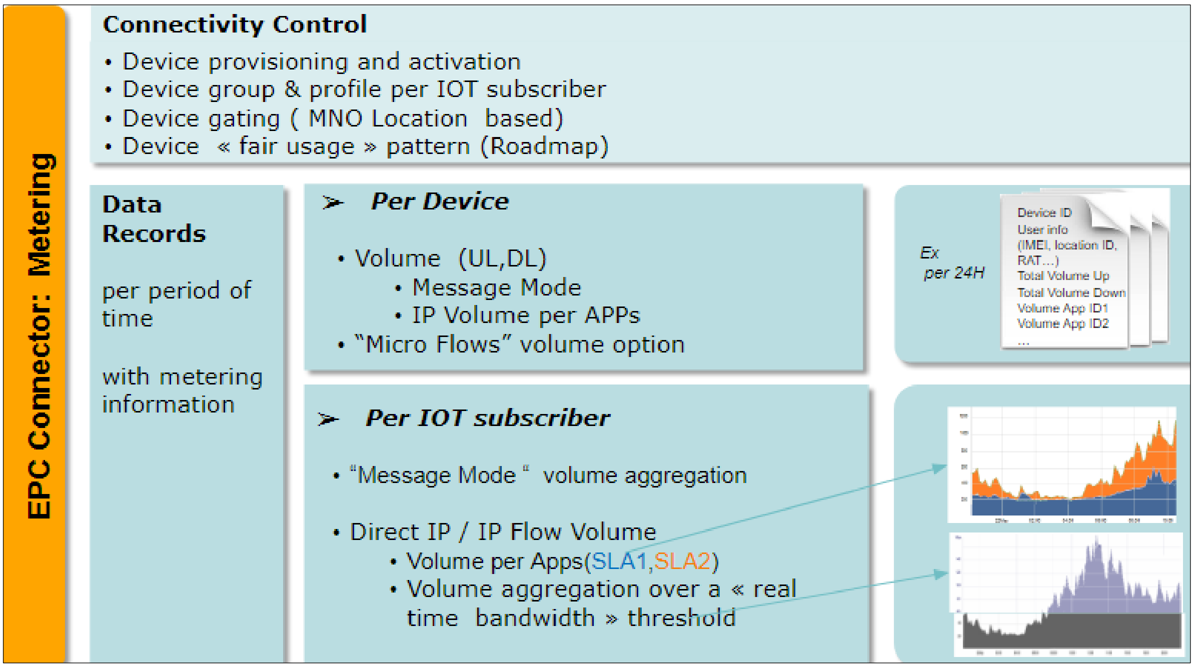


Figure 5 : Device Activation and Metering Functions

EPC Connector allows easy integration of charging records for Cellular IoT devices for both "message mode" and "Direct IP" mode. As shown in Figure 4 and Figure 5, the charging records are generated on a per device and a per IoT subscriber basis, and account both for aggregate volume and microflow records.

ThingPark Wireless billing features are designed from ground up to meet the needs of IoT use cases

2.5 EPC Connector: Policy Enforcement

Policy enforcement for IoT devices is quite complex as there is need to manage enforcement for just not for each device individually, but also at IoT subscriber level, which represents a large group of devices. IoT device traffic patterns can be quite unpredictable at device level but become much smoother and consistent at aggregate level. Policy enforcement is the key to manage congestion in the cellular network and to maintain SLAs for human-centric, mission critical and premium IoT applications. There are many IoT applications which can result in synchronized activity from many devices, hence enforcement needs to be done in real-time to avoid large overheads in the operator network. EPC connector provides several policy enforcement options, such as blocking or shaping, which are outlined in Figure 6.

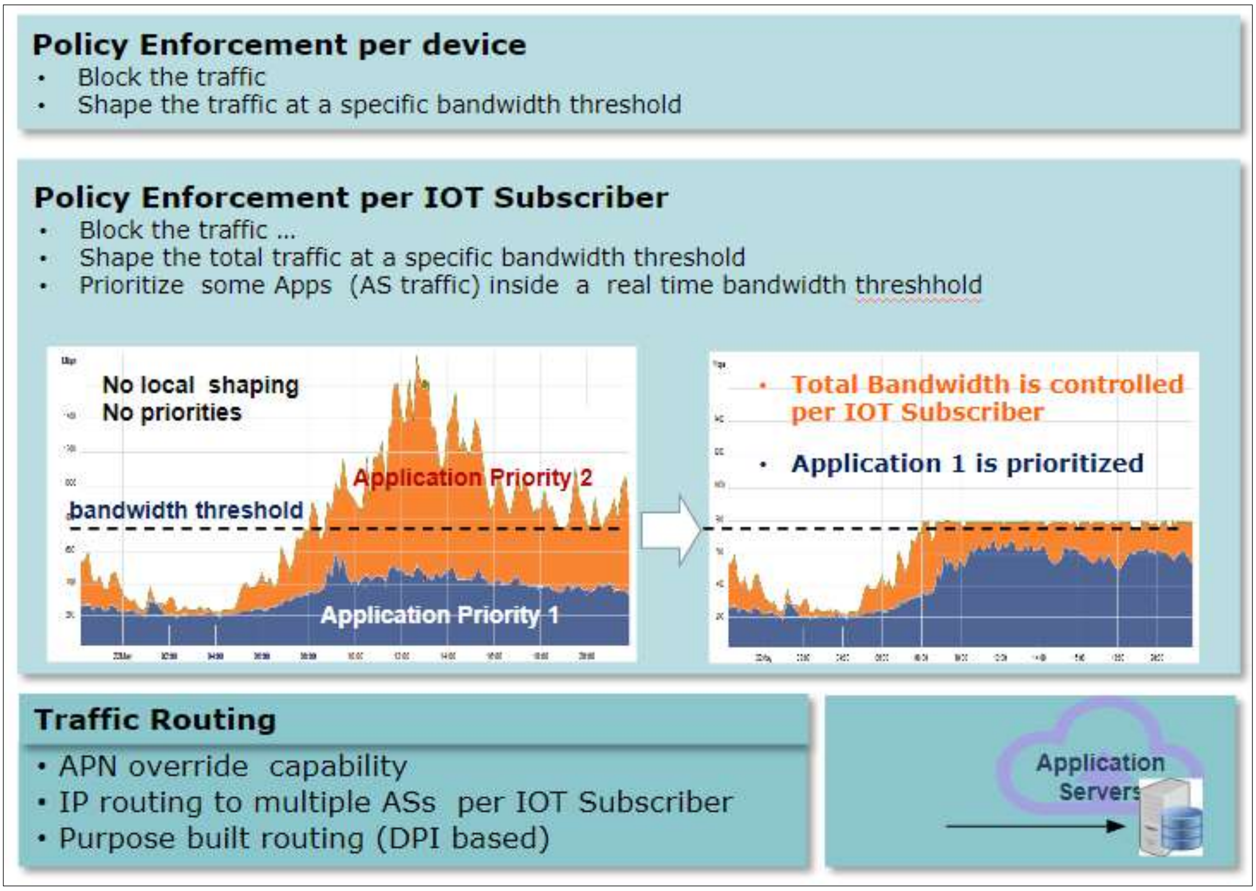


Figure 6 : Policy Enforcement

IoT Policy enforcement to manage group of devices is key to manage congestion and deliver SLA to human-centric or premium IoT applications

2.6 ThingPark X: OSS/BSS APIs for Integration with 3rd party platforms

IoT is synonymous with full automation, and therefore extensive support for APIs is an essential requirement for any IoT platform. APIs are also a way to introduce eSIM/eUICC features and build automated SIM lifecycle management, to scale IoT use cases to billions of devices.

ThingPark Wireless provides 100% API access to all platform features, and ThingPark X extends the scope of APIs to dataflow management, and semantics.

Once connectivity is in place, the value in IoT resides in extracting value from the data collected from end-devices, utilizing various data analytics tools. This often requires decoding and storing the data for later analysis. There is a very vast offering of cloud services and platforms available to process IOT data.

ThingPark X offers the DX API to easily configure dataflows to seamlessly integrate into all market leading IoT data management platforms, such as Microsoft Azure, Amazon IoT, IBM Bluemix, GE Predix, ThingWorks, etc and many other IoT platforms. The API also allows to configure pre-built drivers that turn often complex device semantics into standard XML/JSON documents flexible. The dataflows may also be used to integrate to the operator own data platforms.

Figure 7 shows the architecture of DX API which can be used for the following benefits:

- Subscription & user management
- Offer subscription
- ThingPark Single-Sign-On
- Device management
- Gateway management
- Device data decoding
- Device data forwarding to third party cloud platforms (IBM BlueMix, Amazon AWS, Microsoft Azure)

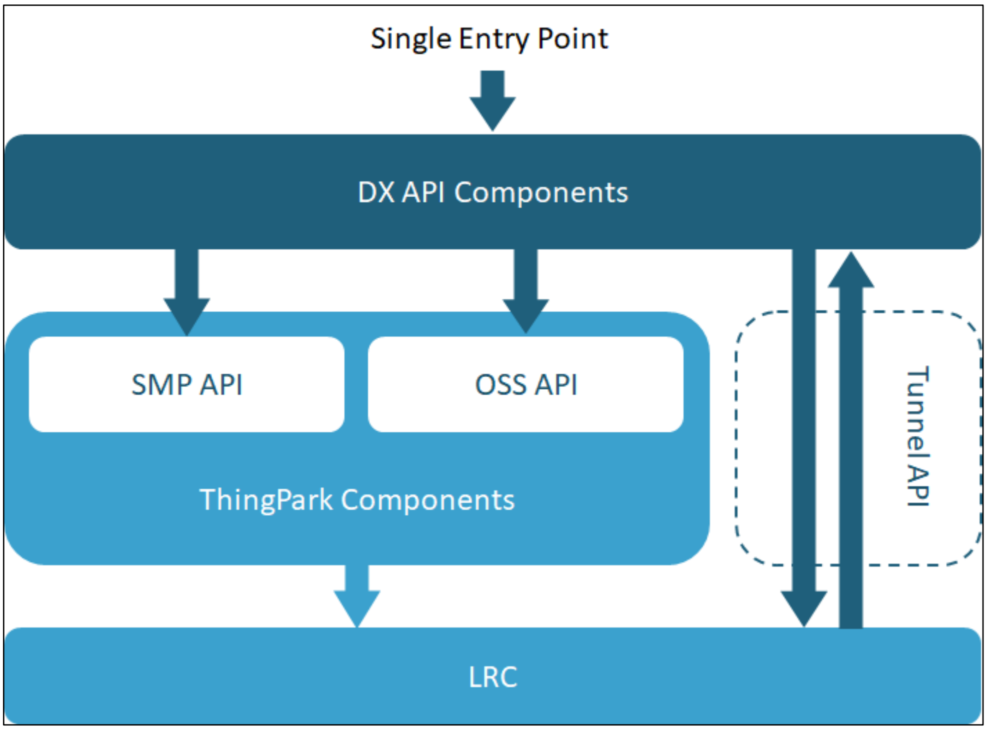


Figure 7 : DX API Integration with 3rd party platforms

Open APIs and modular platform are the key to open innovation and industrialization of IoT landscape

2.7 Summary

ThingPark wireless is carrier grade platform allowing operators to deploy converged IoT services. ThingPark offers:

- Scalability, multi-tenancy
- Choice of licensed cellular (NB-IoT, Cat-M1, Cat-1) and unlicensed (LoRaWAN™) technologies to optimize 100% of use cases.
- Standard integration to any 3GPP network
- Flexible charging and policy enforcement models fully optimized for IoT
- Whole solution addressing dataflow, decoding, and popular cloud platform integration.
- Open ecosystem management via ThingPark market

3 IOT Environment

For the end-user to understand this product, this section provides a quick overview of the concepts related to LTE access for IOT traffic. It is an organized list of terms with their brief explanation which should allow the reader to get a quick understanding of the ecosystem and the role of EPC Connector within ThingPark Wireless.

IOT device

Mostly working with no user interaction, a simple device, generally designed for low power consumption with simple requirements for data.

Such a device is expected to require:

- Small data exchanges from time to time (for instance sending a temperature or measurement to a central system)
- Or bigger, sporadic exchanges of data (image upload, firmware download)

Devices range from low cost, low consumption, high battery life standalone devices to cards embedded in a system with sufficient power and higher networking demand.

AS/Application server

This is any server, hosted remotely, which needs to collect data from many devices, and to send them updates or triggers.

IOT subscriber

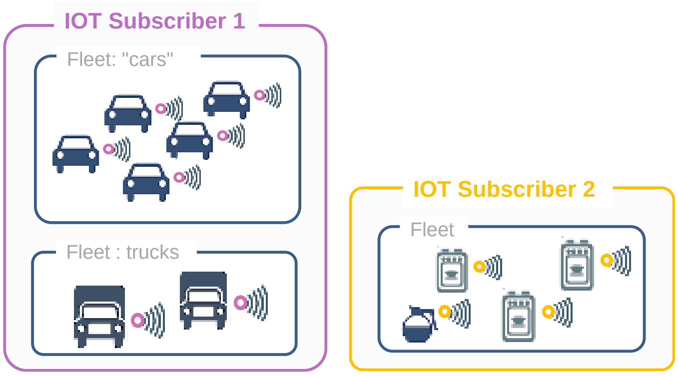
An IOT subscriber is an entity which owns devices and applications servers for a given purpose (locate and track objects/animals, provide alarms on a remote site, measure usage of objects…).

IOT subscribers apply for a connectivity service to get an infrastructure allowing communication between their devices and AS.

Fleet

An IOT subscriber has a fleet of devices, all of them having the same communication and management needs.

An IOT subscriber can have several fleets. Fleet is therefore the granularity of choice for IOT connectivity and services. They are created and managed explicitly in the system.

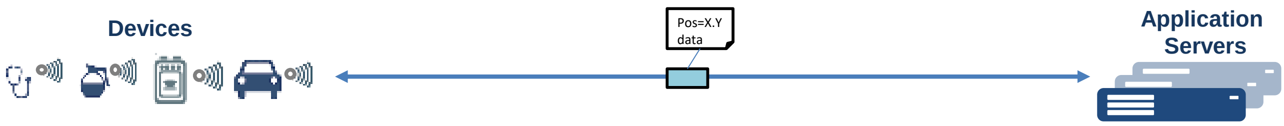


Message Mode

In this document and in the system, message is a small data packet exchanged between the device and the central system.

What makes the message special compared to a basic IP packet for instance is the fact that it is delivered to the AS with additional contextual information from the network. The typical information bundled with a message in uplink is the location of the device when the message was emitted. Note that downlink messages have no bundled information.

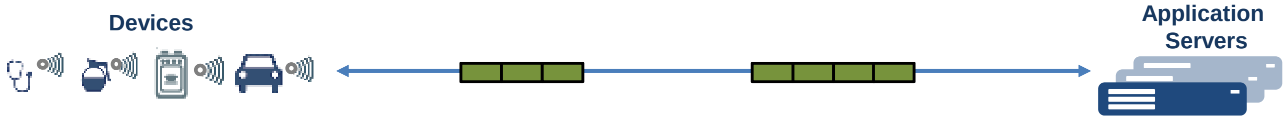
A second important feature for a message is that the delivery is handled by a system (such as ThingPark Wireless) which may store it, duplicate it, and forward it when convenient, with no hassle on the device and server to manage this asynchronous aspect. Recently, 3GPP has introduced the concept of Non-IP Data Delivery (NIDD) to send short messages from Rel13 IoT devices. We transport these short-messages via message mode.



Direct IP Mode

By contrast with messages where relatively heavy processing is done, some types of exchanges require a direct connection to a remote AS, with plain IP.

For instance, a download of firmware is more efficiently managed using the TCP/IP layer. There would be no added value and a scalability issue if all packets were individually enriched, stored and forwarded.



Note that some devices need only messages, some other need direct IP. "Mixed mode" is possible to have both simultaneously.

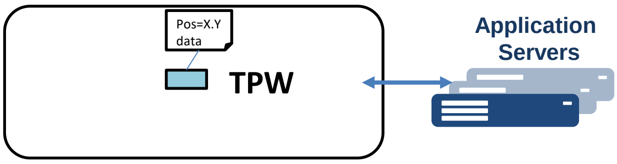
For instance, the device may wake up, receive a message from the application server which requests direct connection, and then start a download in direct IP.

ThingPark Wireless (TPW)

This is the OSS/BSS solution designed to store and forward, for each device, all messages and to bring the tools needed to manage, operate and charge IOT traffic.

ThingPark Wireless allows reaching LoRaWAN™ devices using LoRaWAN™ Network Server, as well as LTE devices though the "EPC connector" defined hereafter. It shows the same API to the AS to send/receive messages from both LoRaWAN™ and LTE devices.

ThingPark Wireless manages "message mode" completely and is also aware of direct IP communications but does not relay direct IP traffic.



EPC

As per 3GPP standard, it's the network part between the device and the Internet which provides IP connectivity with support for roaming. Only LTE access is considered, not 3G or 2G.

EPC comprises of **MME** (mobility and connection management, signalling stratum), **S-GW** (packet relay with mobility support, user stratum), **HSS** (Home Subscriber Server) and **P-GW** (Packet Gateway).

The goal of the EPC is to enable a network of roaming devices and connect it to a remote fixed network (which could be the Internet or a private network).

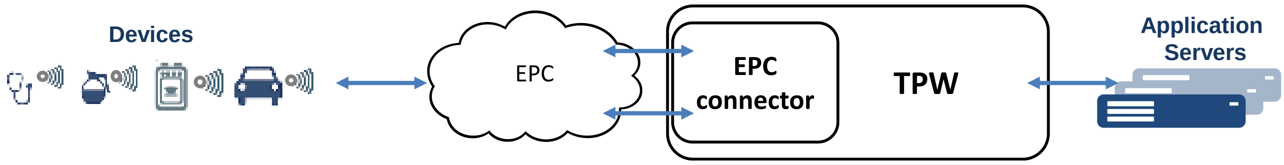
USIM, eUICC/eSIM, IMSI

The EPC identifies any connecting device not by its own hardware id (IMEI) but by a specific id called IMSI.

The IMSI uniquely matches a secret key which is hidden in a piece of hardware (USIM card) or software (eUICC/eSIM). This is an extension of the network inside the device.

EPC connector

As stated in the introduction, it is part of TPW and supports the 3GPP compliant interfaces to the operator's EPC.



These interfaces are usually provided, in the standard 3GPP network architecture, by the following software modules:

- **MTC-LTE-GW** (access gateway for LTE for providing integrated functionality of PGW/PCEF/PCRF/OFCS/OCS)
- **HSS** (device/user database for authorization and authentication)

The EPC connector is therefore split into the MTC-LTE-GW function and HSS function.

Moreover, a **SPR** (subscriber profile repository) function is co-located with the HSS. It contains the provisioned list of devices and the IOT subscriber and fleet they belong to. It is queried by the MTC-LTE-GW at device connection.

APN

In 3GPP, Access Point Name refers to a group of devices in the same network. They have usually a consistent IP range, and their traffic needs to go to the same remote network.

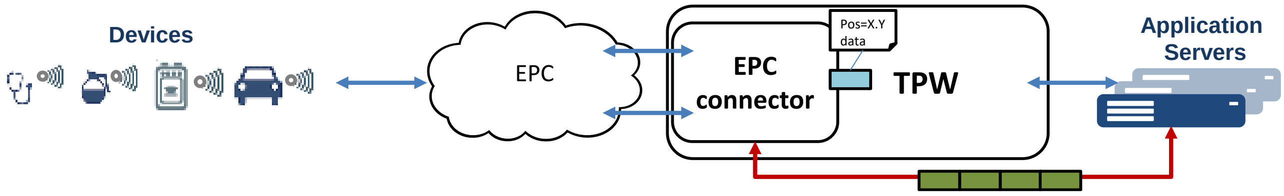
However, APN management is a big constraint: the right APN string must be configured both in the HSS (provisioning) and in each device (personalization) before it first connects.

The MTC-LTE-GW attempts to avoid relying too much on the APN string. Instead, a default value for all devices may be used. The MTC-LTE-GW relies on the fleet concept to manage devices in consistent (and isolated) groups.

Tunnelling

Tunnels allow extending a private network over remote places. They are used to connect devices in each fleet to a remote network. The Internet may be used as a transport while keeping the private network separated.

Tunnelling is used in the solution to enable direct IP for a fleet: the devices and AS are exchanging data as if in a local network. There is no packet "transformation" (NAT or Firewall for instance) needed, and the other networks are fully separated (Internet, other ASes, other devices).



The AS may run some software to terminate the tunnel, or a separate server may be used if multiple AS need to share the connectivity with the fleet of devices.

TP OSS/BSS DX – API

This API refers to OSS/BSS APIs provided by ThingPark Wireless to integrate third party OSS/BSS systems or applications.

Provisioning API

This API takes place between TPW (TWA) and the HSS/SPR to declare devices and tell their fleet.

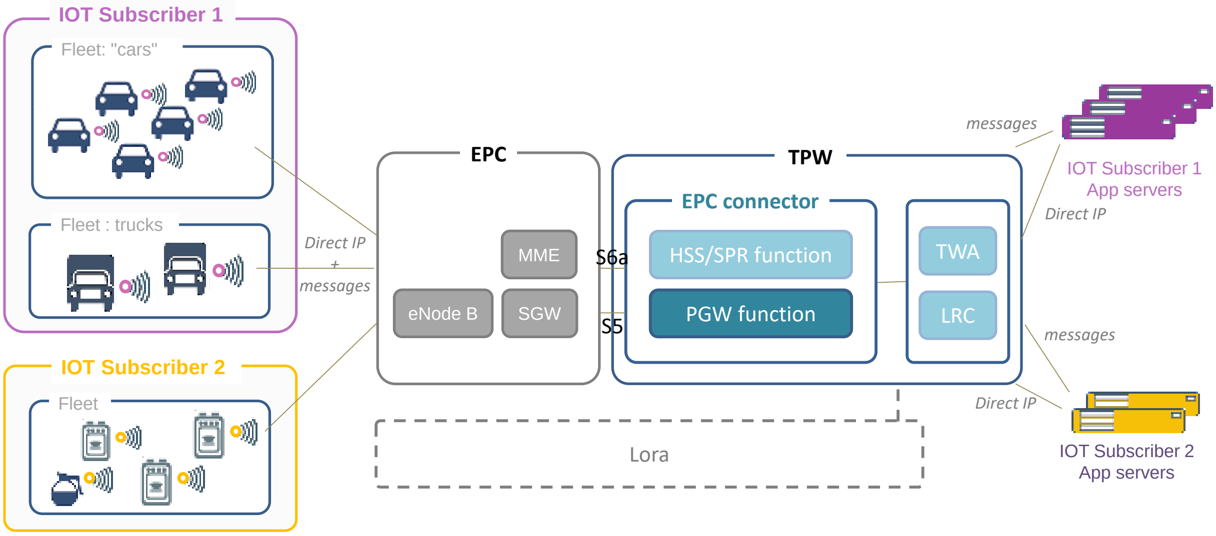
The MTC-LTE-GW gets this information from the HSS/SPR when needed.

Activation / IoT Subscriber API

As the EPC connector is integrated into TPW, a specific API allows activating the fleets: TPW GUI shows the needed properties and accepts user input. In turn, this is translated into API calls to create the fleets inside the MTC-LTE-GW and give their characteristics that the MTC-LTE-GW needs for connectivity, enforcement and charging. Most of the features of the MTC-LTE-GW are driven using "IoT subscriber API" which describes the following:

- The IOT subscriber the fleet belongs to
- The connectivity for the devices – AS direct IP, and for messages
- enforcement rules adapted to IOT traffic for all devices belonging to the fleet
- charging rules adapted to IOT traffic from all devices belonging to the fleet

Ecosystem overview



4 THINGPARK WIRELESS EPC CONNECTOR SOLUTION OVERVIEW

EPC Connector is an addon within ThingPark Wireless that provides insertion into existing 3GPP mobile operator core using standard-compliant interfaces (S6a, S5/S8, T6a) and provides unified device and connectivity management for Cellular IoT devices. EPC Connector enables tighter integration of Thingpark Wireless with 3GPP Core networks. Figure 8 describes how EPC connector interacts with 3GPP network, ThingPark Wireless internal components (LRC, TWA) and Application servers.

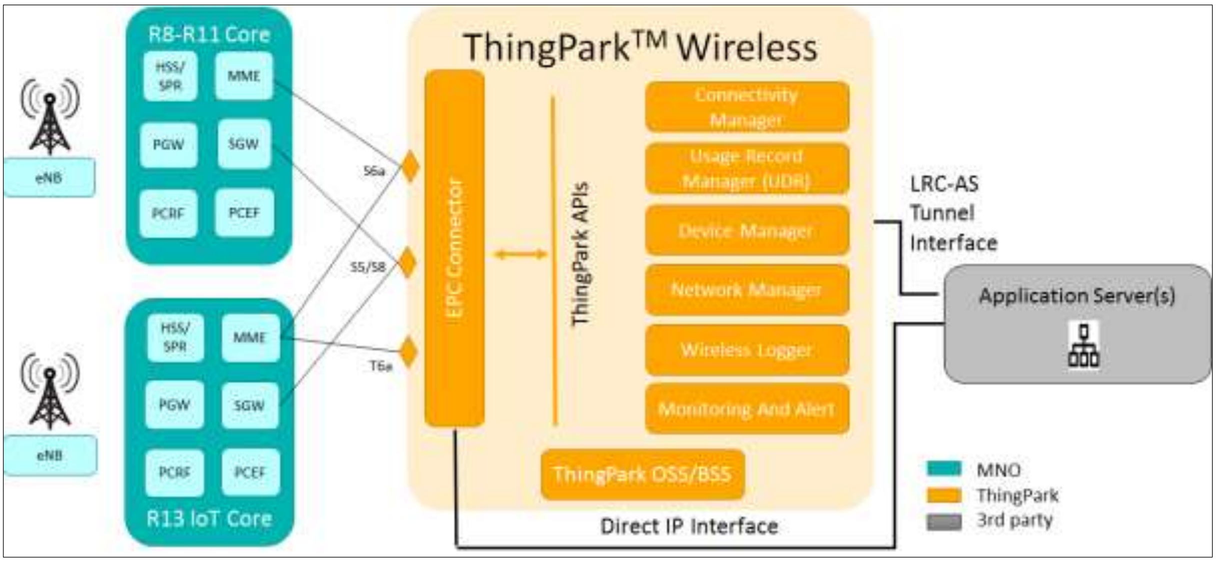


Figure 8 : ThingPark Wireless EPC Connector Overview

EPC Connector interacts with standard 3GPP core network elements using following interfaces:

- S6a: This interface implements the following functionalities of HSS/SPR to interact with MME
  - Unique provisioning layer for 3GPP devices which results in lower IT cost for provisioning and mass insertion of devices
  - IOT-oriented policy and charging provisioning
- S5/S8: EPC Connector implements integrated PGW, charging and policy enforcement module which has following functionalities:
  - Includes specific IoT policies and allows for message oriented communication leveraging unique ThingPark Wireless features and APIs
  - Support for direct IP through private and over the top network for efficient communication
  - Support standard interfaces to the EPC (S5/S8) to enable LTE connectivity

- Provide a means to distinguish the messages and forward them enriched to/from TPW
  - Provide end to end direct connectivity between devices and AS through a tunnel, in a private network
  - Provide metering features to enable TPW to do charging both for message mode and direct IP
  - Provide enforcement for direct IP (TPW may do so for messages)
  - Integrate with ThingPark through APIs
  - It provides integrated functionality of PGW+PCRF+OFCS+PCEF

- T6a: EPC Connector implements functionalities of SCEF/NIDD using this interface.

5 THINGPARK WIRELESS EPC CONNECTOR ARCHITECTURE

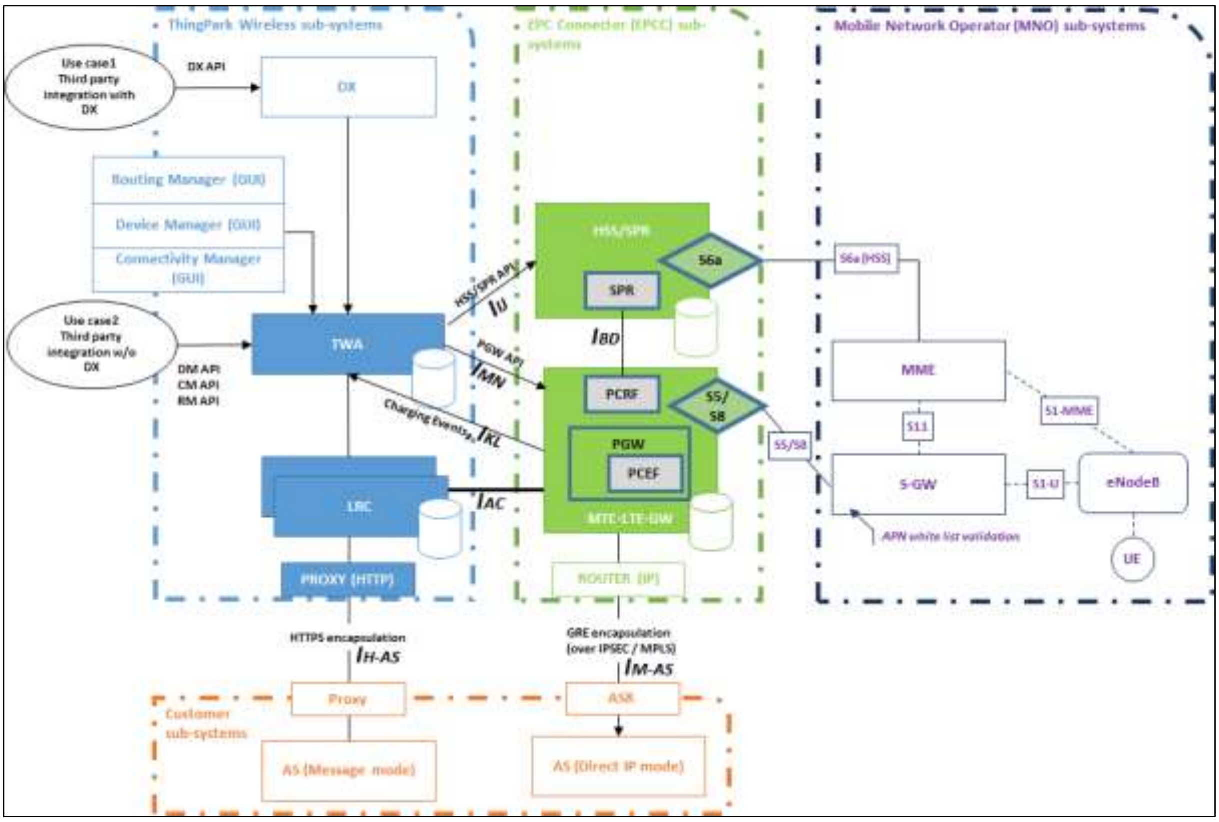


Figure 9 : ThingPark Wireless EPC Connector Interface Description

EPC connector consists of two fundamental modules as shown in Figure 9. One of them is HSS/SPR responsible for providing device provisioning functionality. The other module is MTC-LTE-GW which includes functionality for charging, traffic shaping and policy enforcement. MTC-LTE-GW implements integrated functionality for PGW, PCRF, OFCS, OCS and PCEF. It has the following high-level interfaces:

- **I_{IJ}:** This interface is used for provisioning devices between HSS/SPR and ThingPark. This interface is described in detail in section 5.1.1.
- **I_{BD}**: This interface is used by MTC-LTE-GW to fetch SPR values from HSS/SPR database. This interface is described in detail in section 5.2.
- **I_{AC}**: This interface is based on IEC104 and carries the user payload between MTC-LTE-GW and LRC in "message mode". The description of different modes of connectivity are described in section 6.
- **I_{MN}**: This interface is used to provision IoT subscriber from ThingPark to MTC-LTE-GW. Note, that this API is internal to ThingPark Wireless and EPC Connector and not relevant for customer.
- **I_{KL}**: This interface is used to generate CSV/Kafka based user traffic records. The charging mechanism is descried in detail in section 7.

- **$I_{M\text{-}AS}$**: This interface carries IP packets for IoT device for direct IP path between MTC-LTE-GW and AS.
- $I_{H\text{-}AS}$: This interface carries IP packets encapsulated in LRC-AS tunnel API. It is used for carrying “message mode” payloads between IoT device and AS. This interface is described in more detail in section 5.7.
- S6a: This interface is used by HSS/SPR to communicate with Mobility Management Entity (MME) of the operator network.
- S5/S8: This interface is used by MTC-LTE-GW to talk to Serving Gateway (S-GW) within the operator network. MTC-LTE-GW provides PGW functionality when talking to SGW within 3GPP core network of an operator.

We now describe all the interfaces in more detail.

5.1 I_{IJ}: HSS-LRC Provisioning Interface

This interface is used for provisioning devices within HSS/SPR from ThingPark Wireless. It contains the fields specific to 3GPP security parameters and the SPR fields related to traffic management. We describe below the format of the data structure that is pushed from ThingPark Wireless to HSS/SPR when a new device is provisioned in ThingPark. This interface carries two types of provisioning data:

1. HSS Provisioning
2. SPR Provisioning

5.1.1 HSS Provisioning

A 3GPP standard compliant HSS is used to allow the devices to attach to the network (providing authentication features) and carry data updates during mobility events from one MME to the other. S6a interface is the sole point of integration which is covered with the network, as only LTE access (3GPP Rel 8+) is supported.

HSS complies with the following 3GPP TS:

TS 29.272 - *Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol*	Includes most of the messages between the HSS and MME. A part of this specification is ignored (for ex. 3G access)
TS 23.401 - *General Packet Radio Service (GPRS) enhancements for*	Includes some useful information related to the user’s data to store and exchange between nodes of the network.

Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access

This above list is non-limitative.

HSS is tightly integrated to ThingPark Wireless to provision the right data which is detailed further below.

HSS provisioned information, per IMSI

HSS requires a relatively small set of data to operate. These fields are automatically set by TPW using the provisioning interface.

Note: All this data is related to the USIM and is stored in ThingPark database.

IMSI

It is the Sim card unique identifier. Note that a physical card is not always present in the device, when using a software solution (based on eSIM/eUICC, Universal Integrated Circuit Card)

Ki

Sim card long term pre-shared key

APN

Whitelist of allowed access point names. This information must be set to allow access for the device

IMEI

Device unique identifier

This is a pointer to the SPR Table. Note that an internal binding is done between SPR and HSS using this information

OPid

As the HSS must prove its legitimacy to the device as the known operator, an operator key must be used. This is an index on this secret key

MSISDN

Subscription number for voice services. It is unused and not provisioned as voice is not an IOT service. It is referred here for future use only

AMBR (UL, DL)

This field is used to tell the network which maximum bitrate shall be enforced by the access (MNO) network. Note that the P-GW function does not use this field, which is replaced by shaping information found in the enforcement profile as described in Section 8

Extra field (not usually used in the HSS itself):

Device static IP – IPv4 Optional and *not recommended*

Allows static assignment of the device's IP.

If not present, DHCP is used to provide an IP to the device when it connects, and ThingPark will have to learn it and keep track of its possible changes. As the IPv4 is valid only for a given APN, it is a per APN value.

Note that if an IPv4 is set this way, it must be unique (it means that it should not be part of the DHCP pool) and must belong to the APN subnet. Therefore, caution is needed if this feature is used.

5.1.2 SPR Provisioning

SPR (subscriber profile repository) is an independent device database bundled with the HSS.

The MTC-LTE-GW function must know the device to tell how to manage its traffic, how to charge it, and identify some additional information (barring, and per device additional info), like a name for a group of devices (IOT subscriber and fleet) which is not related to the 4G access.

While the HSS may be left unused (if an external HSS is used), the SPR is core to the EPC connector and must be used for proper functioning of MTC-LTE-GW.

IMEI – *key*

Device Unique Identifier

Fleet – string(20) - mandatory

IOT Subscriber – string(20) - mandatory

Currently, in TP 5.0 there is one to one mapping between IoT subscriber and Fleet objects. Hence, throughput the document, we use Fleet and IoT subscriber synonymously and ThingPark GUIs/APIs always refer to IoT Subscriber. However, in future releases of TP 5.x+, this restriction will be removed and one IoT subscriber could have several fleet objects. This would allow grouping of devices within same IoT subscriber with same routing profile/connectivity plan settings. But the distinct groups have slightly different settings for routing profile/connectivity plan. This would allow more automated provisioning of a group of IoT devices sharing similar characteristics using OSS/BSS APIs of ThingPark and significantly bring down device provisioning costs incurred by an IoT subscriber.

Barring int – string (2) - mandatory

It is used to tell if the user should be temporarily blocked. 0 (default) = no barring. 1 = full barring.

It may be extended in the future to selective barring, for instance prevent roaming access. (2 local only, 3 local and partners). However, such domains are not defined in the system yet.

Network context name override (string 20) - Optional

This is used to map the device to a specific network context (connectivity to the right remote AS)

The expected usual behavior is that the MTC-LTE-GW function evaluates the fleet field. The MTC-LTE-GW function contains a fleet object which tells the network context to use for devices belonging to this fleet. However, setting this field for a device within HSS provisioning will give the MTC-LTE-GW function the name of the network context to use directly.

Note: see references [6][7] on network contexts and their usage.

***Device Policy override*/profile** (string 20) - Optional

This is used to associate the device to an enforcement profile (per device limitations).

The expected usual behavior is that the MTC-LTE-GW function evaluates the fleet field. The MTC-LTE-GW function contains a fleet object which tells the enforcement profile to use for devices belonging to this fleet. However, setting this field for a device within HSS provisioning will give the MTC-LTE-GW function the name of enforcement profile to use directly.

Note: see references [6][7] on enforcement profiles and their usage.

<u>Contextual enrichment</u>:

Optional: allow enriching information into data records produced by the MTC-LTE-GW function.

In MTC-LTE-GW function, they are simply copied as metadata. They are appended to messages and DDR. See PGW function introduction [7] for details about "messages" and "DDR".

FreeField – String(20) - optional

Free form field which is not used in the current release

5.2 I_{BD}: MTC-LTE-GW – HSS/SPR Interface

The interface, I_{BD} is used by MTC-LTE-GW to fetch SPR values from HSS/SPR database. This interface is internal to EPC connector and not exposed to other components of ThingPark Wireless or other 3GPP network elements. The details of the fields used in this interface are specified in section 5.1.2.

5.3 I_{AC}: IEC104 MTC-LTE-GW LRC Interface

This interface is based on IEC104 and carries the user payload between MTC-LTE-GW and LRC in "message mode". The user payload in UL is enriched with 3GPP metadata shown below which is extracted when interacting with the operator network.

Field	**Size**	**Presence**	**Signification**	
Version	1 Byte	**M**	Current message format version (1=first version)	
Message Type	1 Byte	**M**	Type of event :	
			0	IP Data
			1	Session Creation
			2	Session Deletion
			3	Session Modification
Cause	1 Byte	**M**	Cause of session deletion, provided by MTC-LTE-GW only. Must be 0 in all messages from LRC	
EBI	1 Byte	**M**	Extended bearer identification, provided by MTC-LTE-GW.	

			Can be 0 in all LRC messages.
LRC Address	4 Bytes	**M**	LRC IPv4 Address. Must be filled in both directions.
UUID	8 Bytes	**M**	Sensor identification = IMSI BCD encoded (TS 29.274 / E.212) Must be filled in both directions.
Sensor Address	4 Bytes	**M**	Sensor IPv4 allocated address. Must be filled in both directions.
APN Id	4 Bytes	**M**	Apn identifier declared in APN table and able to identify APN. Can be 0 in all LRC messages.
TAG Name	8 Bytes	O(*)	Name of the current Tag filled by zero if name size < 8 bytes
TAG Length	2 Bytes	O	Tag length in byte
TAG Value	N Bytes	O	Tag value of Tag length bytes (32 bits aligned)

Presence:

M **:** Mandatory in both directions

O **:** Optional.

(*) : When Message Type is IP Data, at least one tag named "IPDATA" is present.

From LRC to MTC-LTE-GW only IP Data type messages are allowed. These messages contain a unique IPDATA Tag. Tag allows transmitting LRC some PGW session information which is the metadata collected during interaction with operator's 3GPP network. Session related tags are presents in all messages, but only IP Data messages contain IPDATA Tag.

TAG list is dynamic. Number of tags is not fixed since it depends from variables present in MTC-LTE-GW USV Data base.

The following table shows current MTC-LTE-GW tags:

TAG Name	*Type*	*Usage*
IPDATA	Binary buffer	IP Data including IP Header and TCP/UDP Headers
MSISDN	String number	MISDN number E.164
IMEI	String number	International Mobile Equipment Identity
RAT	String	Radio Access Type
CELLID	String number	Cell identification
CMCCMNC	String number	Cell MCC/MNC
CELLTAC	String number	Cell Tracking Area number
SERVNET	String number	Serving Network MCC/MNC

5.4 I_{MN}: IoT Subscriber Interface

This interface is used to provision IoT subscriber from ThingPark to MTC-LTE-GW. The details of IoT Subscriber creation within ThingPark are internal to the product.

5.5 I_{KL}: Kafka UDR Interface

This interface is used to generate CSV/Kafka based user traffic records. The charging mechanism is descried in detail in section 7.

5.6 $I_{M\text{-}AS}$: Direct IP Interface

This interface carries IP packets for IoT device for direct IP path between MTC-LTE-GW and AS. It is also referred to as SGi interface in 3GPP terminology. However, unlike SGi interface which provides connectivity to Internet, this interface provides connectivity to Application Server Routers (ASRs) over IPSec/GRE tunnels for the device to connect with Application Servers.

5.7 $I_{H\text{-}AS}$: LRC-AS Tunnel Interface

This interface carries IP packets encapsulated in LRC-AS tunnel API. It carries "message mode" payloads between IoT device and AS. It carries the following metadata that is sent from MTC-LTE-GW:

- MSISDN: MISDN number E.164
- IMEI: International Mobile Equipment Identity
- RAT: Radio Access Type
- CELLID: Cell Identification
- CMCCMNC: Cell MCC/MNC Information

- CELLTAC: Cell Tracking Area number
- SERVNET: Serving network MCC/MNC

6 Modes of Connectivity

EPC Connector interacts with Application servers (AS) in three modes which are configured in the connectivity plan (see Section 13.5):

1. **Mode 1: Message mode:** In this mode, infrequent UDP messages (on a specific UDP port) from the devices are forwarded to LRC within ThingPark which it forwards these to Application servers. In this mode, we leverage Thingpark Wireless' capabilities for charging, traffic shaping, wireless data logging and other features. 3GPP Non-IP Data Delivery (NIDD) is also implemented using this mode.
2. **Mode 2: Direct IP mode:** In this mode, IP traffic from EPC connector is directly routed to Application servers. However, the metadata regarding the traffic characteristics is still forwarded to ThingPark Wireless Application (TWA) Usage Data Record (UDR) for charging purposes. However, the microflow reports (which contains statistics of the Direct IP transfer) are still visible in wireless logger without the actual data.
3. **Mode 3: Mixed mode:** In this mode, both the message mode and direct IP mode are used for communication between devices and AS.

6.1 Mode 1: Message mode

In this mode, every packet arriving at MTC-LTE-GW is forwarded to LRC using I_{AC} interface (described in section 5.3), which then forwards it to Application Servers (AS) based on the AS configuration inside ThingPark Wireless. MTC-LTE-GW appends the following LTE metadata to each packet:

- MSISDN: MISDN number E.164
- IMEI: International Mobile Equipment Identity
- RAT: Radio Access Type
- CELLID: Cell Identification
- CMCCMNC: Cell MCC/MNC Information
- CELLTAC: Cell Tracking Area number
- SERVNET: Serving network MCC/MNC

This mode is currently applied only for NIDD or short-messages from end-devices which are contained within UDP packets. LRC and AS communicate with each other using LRC-AS tunnel API which has LTE metadata above appended to it. The details of LRC-AS tunnel API are provided in [10].

The charging and traffic shaping in message mode is done by TP Wireless and follows a model like that of LoRaWAN™ messages coming from LRR. MTC-LTE-GW identifies the traffic in "message mode" by using "source port" of UDP messages in uplink, whereas using the same “destination port” of UDP messages in downlink. The UDP port configuration is carried out in routing profile configuration which is described in section 13.7.

Figure 10 shows the message mode architecture and how packets are transported from the end-device to Application Server (AS).

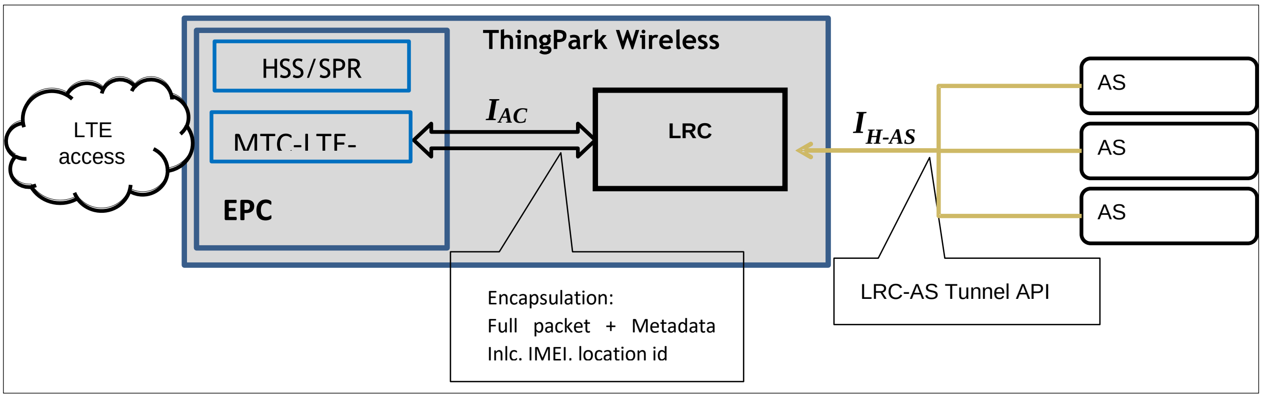


Figure 10 : Message mode

6.2 Mode 2: Direct IP

In this mode, MTC-LTE-GW forwards the IoT device traffic directly to AS, thus performing « over the top » routing. Direct IP mode is useful for sending large volumes of traffic (for example video feed from security camera or firmware upgrade) directly from MTC-LTE-GW to AS without going through ThingPark Wireless. However, MTC-LTE-GW in this case sends metadata related to charging events using interface, I_{KL} in terms of microflows, device data records (DDR) and fleet data records (FDR). It receives configuration of traffic enforcement from ThingPark Wireless via IoT subscriber API, I_{MN}. Figure 11 shows the flow of traffic for « Direct IP » mode.

This mode is also useful to provide flat IP connectivity between the AS and devices. The communication between MTC-LTE-GW and AS happens with an intermediate entity, Application Serving Router (ASR), which talks to MTC-LTE-GW via IPSec/GRE tunnel. This is a mandatory component which terminates the tunnel between the MTC-LTE-GW and the AS local network to enable over the top, private network and is also useful to protect ASs from Internet attacks.

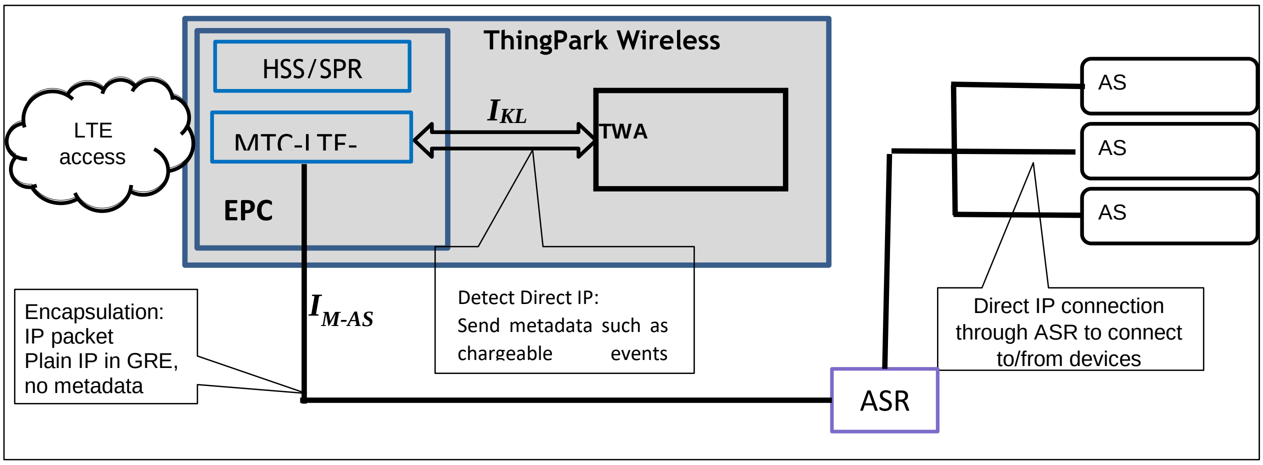


Figure 11 : Direct IP mode

6.3 Mode 3: Message mode + Direct IP

There are use cases when the device wants to do both “message mode” and “Direct IP” at the same time.

In mixed mode, the “message mode” traffic is filtered on MTC-LTE-GW using “specific” UDP port. The “specific” UDP port is configured using IoT subscriber interface, ***I_{MN}***. This allows simultaneous support of both “message mode” and “Direct IP” mode. The UDP message below illustrates an example to identify a message towards a service on a given AS:

- IP:
    - device IP
    - AS IP
    - **UDP:**
        - Device port = Service reserved port -> 333
        - AS port = Service on the AS, as for direct IP.

This makes a standard UDP socket, easily created on the device which allows transporting any message as UDP payload.

This uses a “specific” UDP port on the device (as source port for outgoing packets and listening port for incoming packets). This “specific” UDP port can be configured by AS routing profile (see section 13.7), using IOT Subscriber API, ***I_{MN}***.

7 CHARGING

ThingPark wireless platform elements (LRC in particular) have the capability to charge for IOT traffic in the form of messages as they are stored and forwarded there. For “message mode” traffic, charging is carried out directly by ThingPark Wireless (TPW) by treating all traffic in “message mode” similar to LoRaWAN™ messages. However, for “direct IP” traffic, as actual packets do not transit through TPW, MTC-LTE-GW sends charging information via Kafka UDR Interface, I_{KL}. The product enables different charging strategies for IOT subscribers at distinct levels; however, all of them are optional tools to build rich offers.

Simple volume charging:

Based on number of messages and volume, this is the basis for messages (common approach for LTE or LoRaWAN™ access). This applies both to messages and direct IP traffic.

Event charging

An event is defined as IP activity within time and volume limits (a “microflow”). For direct IP, this mimics the simple charging where the event of some IP activity is charged for.

Instead of message per message, a certain amount of traffic volume is reported in the event and may be charged. For instance, a device with usually low traffic will cause generation of one microflow report per hour (if event time trigger is set to one hour), but if the device sends a large image, an event will be generated as soon as the traffic volume exceeds the value defined as event volume trigger.

Group charging:

IOT services use fleets of devices and not a single device. This very specific property in the service deserves a different charging mechanism:

- To come up with innovative billing plans
- To favor some good practices
- To characterize the traffic more easily than for individual devices (leveraging the smoothing effect of the law of big numbers)
- To better account for the effective cost to the network infrastructure: For instance, having all devices sending a message in the same minute is probably charged more than if the traffic is evenly spread over the entire day.

This impacts the sampling requirements. For direct IP, the MTC-LTE-GW must provide a “real time” measurement to accurately measure group usage. Note, that actual rating and billing can be done using higher level criteria, such as “flat rate” where a unique price is demanded per month, regardless of actual number of messages. This is out of scope of this document. However, EPC Connector provides all the necessary metrics for an operator to roll out group rating/billing for its IoT subscribers.

7.1 Flow-based events

7.1.1 Introduction

Direct IP mode allows the devices to start a communication directly with an application server via MTC-LTE-GW (thus bypassing the ThingPark Wireless IOT OSS/BSS/mediation).

IOT devices are not expected to generate substantial amounts of traffic for long periods of time, but it still happens from time to time for firmware downloads and data file uploads. Packet per packet reporting would not make much sense in this case.

The MTC-LTE-GW does a tailored reporting by detecting the real-time activity using volume triggers, and report near-real time aggregate reports ("microflow reports") to ThingPark Wireless Application (TWA).

7.1.2 Microflow event triggers

A "microflow" is defined in the MTC-LTE-GW as a chunk of traffic, defined by the following triggers:

Threshold

The typical volume granularity to report.

Typical value: 100KB

Example: If a device emits 150KB in a short timeframe, a first event is detected as soon as the first 100K are reached. The remaining 50K will be collected in a second event

Maximum inactivity time for last record

The time after which, if activity has been detected, no matter whether any threshold was reached, an event is reported.

Typical value: 120 sec.

Example: If a device emits 50KB in a short timeframe, and then stops, the flow will be emitted as soon as the 120 secs are reached from the beginning of the activity.

Minimum time between 2 records

A timer to avoid an overload of data records if a high datarate is sustained by a device for an extended period.

Typical value: 20 secs.

Example: If a device reaches 150 KB in less than 20 secs, the next data record will be issued right after 20 secs, reporting the whole volume exchanged during these 20 secs even though the threshold was exceeded. Figure 12 shows the different examples of microflows.

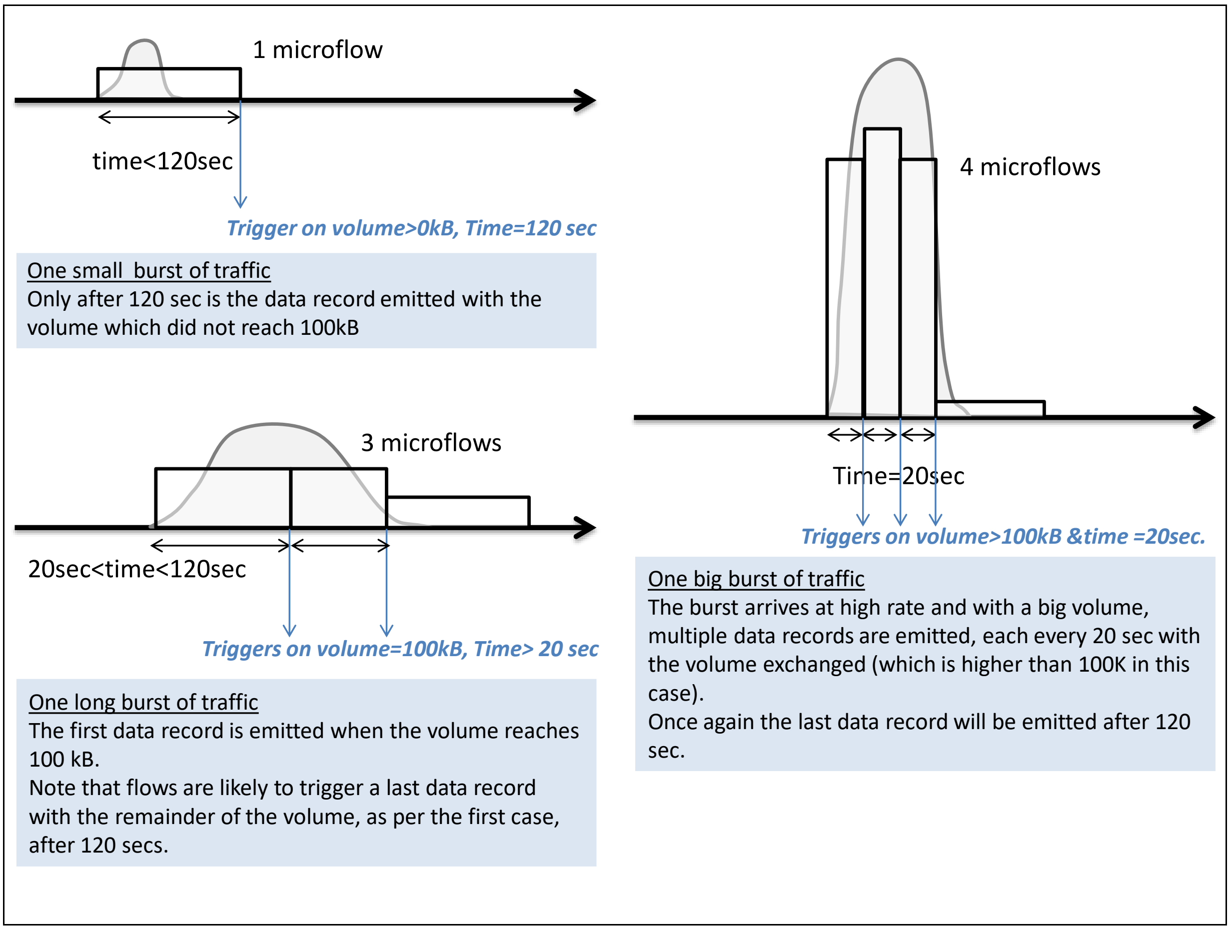


Figure 12 : Examples of different microflows

Values for flows triggers are provided through the IoT Subscriber API so they are the same for all devices of a given fleet.

7.1.3 Usage

One event is issued as soon as a flow is detected.

The event can be sent in real time with the device ID and timestamp, the flow event dimensions (the actual volume and duration). It is also enriched with metadata such as: IMSI, Cell ID and so on.

These events are sent to TWA UDR via Kafka and TWA aggregates these records when it generates UDRs.

Events are also aggregated and reported on a regular basis (not real time), with total volume and number of events. This is the default means to export charging records, which is described further in section below.

7.1.4 Device and Fleet data records

A device will issue / receive messages as well as direct IP translated into flows events. Additionally, it will also incur signalling messages on the MTC-LTE-GW during its connection time. All of this is potentially chargeable information which is tracked on a regular basis. Internal counters are updated when:

- an event is seen online for a device
- a message is seen online for a device
- signalling messages are received or sent by the MTC-LTE-GW for a device

The following information is also gathered:

- Total volume / number of packets for packets in “message mode”, in each direction (uplink/downlink)
- Total volume / number of events for direct IP traffic, in each direction (uplink/downlink)
- Number of GTP-C signalling messages sent and received

They are aggregated twice as described below.

7.1.5 Device Data records

The aggregation is done in two ways:

- per device
- per period (typically 24 hours)

The period is configurable in the product (for default configuration),

The IoT Subscriber API allows setting specific values per fleet if needed.

Data records are organized in files in CSV format. They can also be exported via Kafka in TP 5.x+.

Device is the key entry for each data records (one line per device)

They have additional specific fields: device identification (key), fleet, IOT subscriber, last location

7.1.6 Fleet Data records

The aggregation is done once:

- per fleet
- per period (typically 15 mins)

The period is configurable in the product (for default configuration),

The fleet management API allows setting specific values fleet per fleet if needed.

Data records are organized in files in CSV format.

Fleet is the key entry for each data record (one line per fleet)

7.1.7 Data records export

CSV files are directly available to be retrieved and managed on a regular basis (using scp, ftp) from within EPC Connector virtual machine.

Kafka is also provided as an alternative transport means to exchange data records between EPC Connector and ThingPark Wireless Application (TWA).

ThingPark Wireless servers import this data and create usage data records (UDRs) out of it.

7.1.8 Virtual Lines Data records

7.1.8.1 Overview

A virtual line (Vline) is a resource on the MTC-LTE-GW allowing real time bandwidth measurement for a group of devices. Therefore, a virtual line is typically mapped to a fleet of devices, or to the different fleets of an IOT subscriber altogether.

Virtual line and IOT subscriber association to it is done from the IoT Subscriber API.

Virtual lines come with a threshold of throughput which is compared with the real time throughput going through the Vline: the sum, at a given instant, of all devices communicating at the same time.

The comparison is done based on a leaky bucket algorithm.

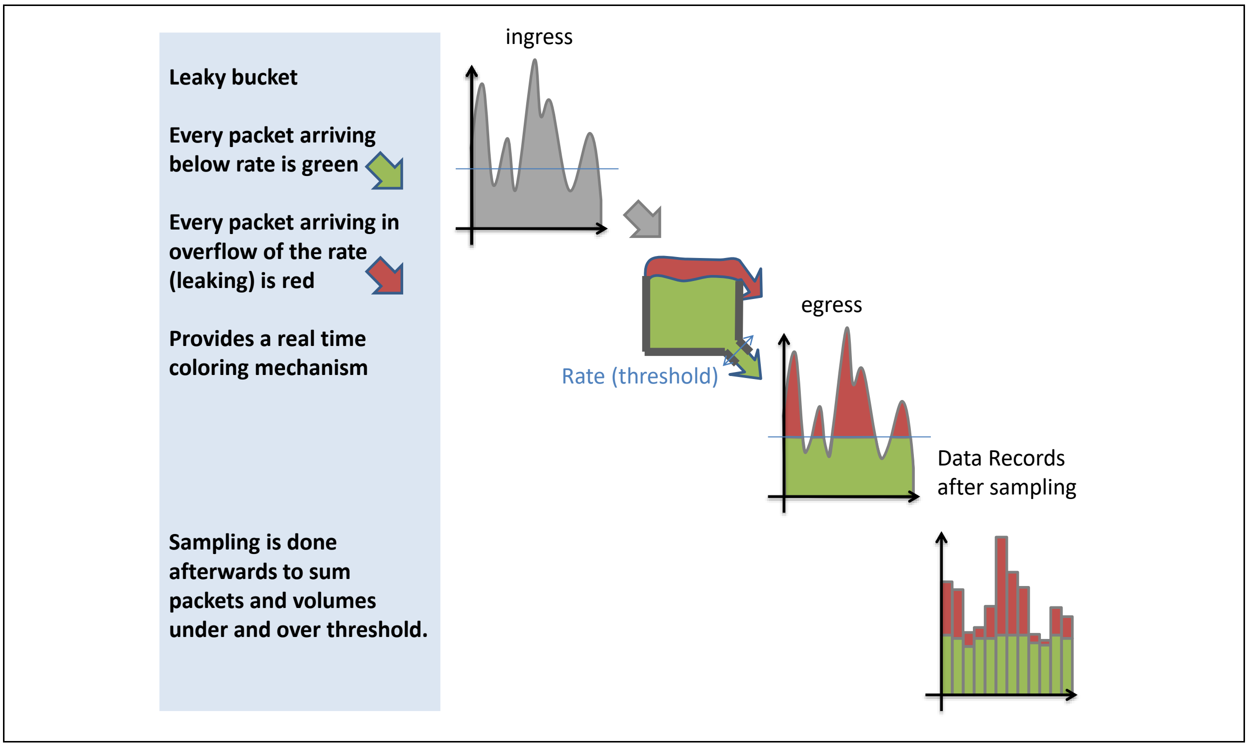


Figure 13 : VLine leay bucket algorithm

Virtual line data records show which share of the traffic (in volume) was exchanged below the threshold and which share was exchanged above. In the leaky bucket, all packets arriving in excess to the rate are marked accordingly and counted in their dedicated counter. Uplink and downlink traffic flows are processed separately.

The usage of the Vline data records is to allow a given subscriber, for its fleets, to do unlimited volume provided that the global traffic of the fleet does not exceed a given threshold. Beyond this threshold, traffic is charged per volume at a higher rate. For instance, a fleet of security devices may upload small data (about their state, and a photo occasionally). This is done with a flat rate. But in the unlikely event that some incident happens they may upload video content in a synchronous fashion. This will be charged different as it exceeds the basic rate.

Note that Vlines are used for policy enforcement for a group of devices within a fleet/IoT subscriber as well – see section 8 on policy enforcement.

7.1.8.2 Vlines Data Record content

They are issued every 15 mins (default value, configurable globally).

They provide the following data:

- Vline ID, Sampling duration and the time stamp
- Packets and volume over threshold; uplink and downlink
- Packets and volume under threshold; uplink and downlink

8 ENFORCEMENT

The enforcement for "message mode" traffic is carried out directly by ThingPark Wireless using the traditional connectivity plan settings. However, for the "Direct IP" mode, the MTC-LTE-GW enforces policies at two levels.

- At device level:
    - A profile is applied, which is preconfigured in the MTC-LTE-GW.
    - It tells limits to enforce to the device's traffic (rates, blocking).
- At IOT subscriber or fleet level:
    - If associated to a VLine, rate limiting can be applied globally.

8.1 Device enforcement profile

An enforcement profile is a preset, which is created through the fleet management API, to do the following:

- Block messages or not
- Block direct IP or not
- Rate limit per device direct IP: uplink limit
- Rate limit per device direct IP: downlink limit

The MTC-TLE-GW includes several rate limiters per device to enable this control. Note that packet out of the allowed rate are simply dropped.

Note, that the MTC-LTE-GW provides only "rate limiting" capability and not traffic shaping capability. Therefore, it never delays packets or messages.

8.2 Vline shaping

For IOT subscribers or fleets which have been associated to a given virtual line, a max rate can be configured to enforce traffic rate limits. The same VLine object is used for both charging (see section 7) and enforcement. This max rate is to be configured above the threshold for charging.

The global traffic associated to all devices belonging to IOT subscribers within a given Vline is rate limited globally. Only "direct IP" traffic is considered as it is expected that messages don't consume lot of bandwidth due to their small IoT payload and are possibly not resent if lost.

Moreover, the LRC in TPW has its own limits for messages which are enforced using connectivity plan settings.

The virtual line has the following characteristics:

- Virtual line ID
- Rate limit up for direct IP
- Rate limit down for direct IP

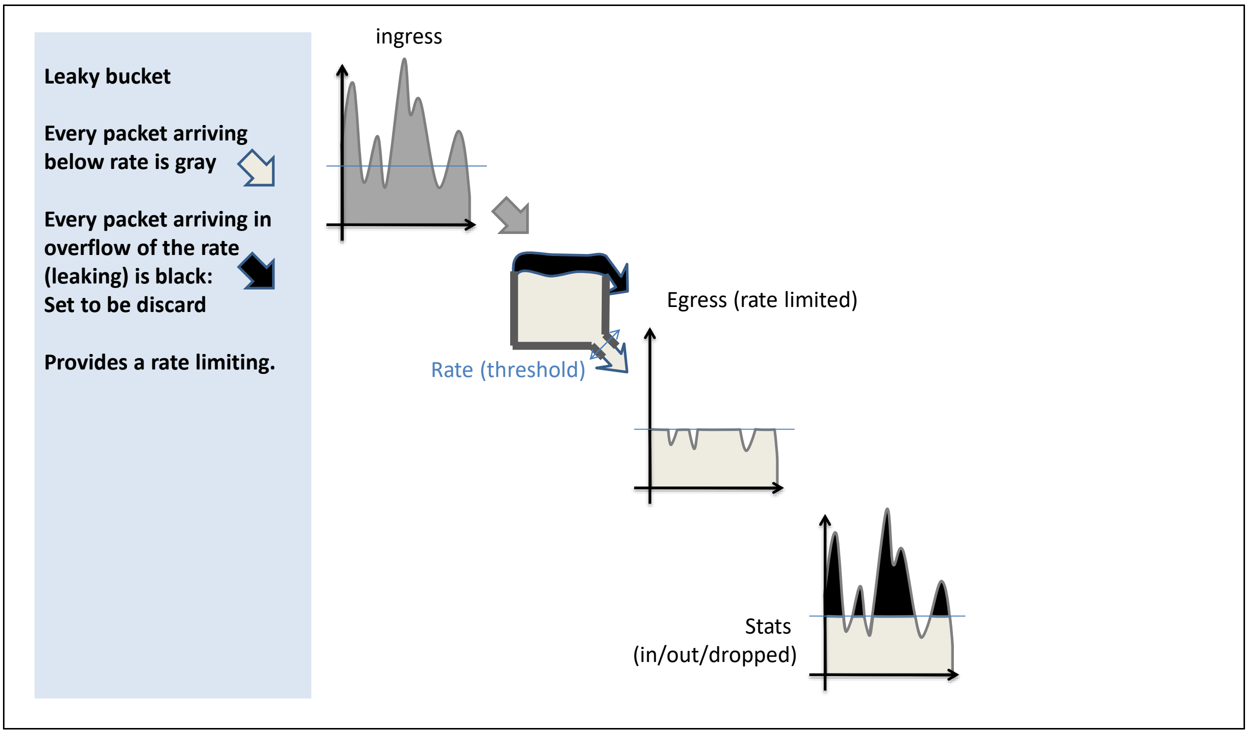


Figure 14 : VLine traffic enforcement

9 SECURITY

Security in the EPC connector is ensured through different means:

- Access control based on usual Linux best practices, using different accounts and limiting access to authorized personal only. This mitigates most of the risks of misuse or unauthorized modification.

- Location of the components behind a firewall to restrict protocols to only the required one.
  - By network design, the HSS function is accessible only from the core network and ThingPark Wireless, and never exposed to any untrusted network.
  - The PGW is recipient of all the traffic to / from customers and devices which is carried over the internet. A firewall is required for protection from direct attacks coming from the internet.

- Cryptography for all secrets. HSS must use and manage several secret keys (per operator and per device). None of these keys are accessible from anyone as they are stored encrypted, the master key for their protection being deeply hidden in memory and accessible to no one. No option is given to read any key, and transport of any key to the HSS is done remotely, through an encrypted API.

- Protection of the Internet from the IOT devices: IOT devices are often low-end products, some of them showed strong vulnerabilities in the past, being turned into botnets. The EPC connector provides two ways to cope with such threats:

  - The traffic of all devices is tunneled up to the end-customer Application Server, which means that a corrupted device cannot reach any other target than its customer's legitimate network.
  - All incoming and outgoing traffic is monitored by the EPC connector which may be used to detect and block any malfunctioning device, at device level and at group level. Rate limiters can also be used as a means of protection.

- Confidentiality for the data going from/to the devices, to the application server is not in the scope of the EPC connector. However, for such requirement, having an end to end protection is most often required and already put in place. It is advisable for customers to implement such mechanism this way (note that anyway GTP does not encrypt the traffic either). Future versions of the EPC connector may provide additional means for very low-end devices which cannot perform any encryption. IPSec can also be used as one of the options to connect Application servers (ASs) from customer premises to PGW.

10 SCALABILITY

A single system in a virtual machine can easily scale to several Gbps and multimillion of devices (depending on the hardware it is running on and the availability of state of the art virtual acceleration techniques like SR-IOV). A key point is that the number of connections per second is not the limiting factor, nor is the number of concurrent connection as it has been with legacy systems.

If more capability is required on the MTC-LTE-GW function, adding servers is done easily. The provisioning is centralized, each server gets its data from a centralized database, and load balancing is achieved on a per device basis. A DNS is usually used to spread the load. HSS nodes can also be added, which requires a Diameter Routing Agent doing the load sharing.

11 HIGH AVAILABILITY

All components of the EPC connector are highly available and switch over is transparent to the access network, which means that devices do not have to reconnect if ever the HSS function or PGW function fails. This is critical to avoid a rush of connections which may overload the 3GPP Core Network.

The EPC Connector components work in an active/standby cluster; the standby server is synchronized in real time with the active server.

When a failed server recovers (for instance upon a reboot), it is unavailable until the resynchronization process has finished, and it has all the states of all devices in memory. This is done automatically and has no noticeable impact on the active node (as states are exchanged with lower priority, this does not block new connections or mobility events). The global cluster is up during the entire process.

12 Disaster recovery

If for any reason, a full cluster fails and is completely lost, then recovery procedures are run:

- The system is first restored/reinstalled from a saved image; this allows setting up a new empty cluster very fast.
- All data is safely stored within ThingPark Wireless and loaded again. A manual procedure allows loading the required data in bulk (data is duplicated in the 2 nodes along the way)

The second step is manually triggered, and no device can reconnect before it is loaded in the system again.

The cluster starting fresh again announces this event to the network so that all previous contexts prior to the failover are flushed immediately.

13 EPC CONNECTOR CONFIGURATION, PARAMETERS AND ATTRIBUTES

This section gives an overview of all the necessary changes required on HSS/SPR, MTC-LTE-GW and ThingPark components (TWA, LRC, etc) for Cellular IoT devices to be supported on ThingPark Wireless.

13.1 TWA configuration for EPCC provisioning

The HSS, SPR and PGW provisioning information from TWA are all performed on master HSS servers, this includes:

- For HSS
  - The provisioning of the devices, with either SPR data or HSS/SPR data (following operator settings) – provisioned on Device creation
- For PGW
  - Fleet – provisioned on Device Manager subscription
  - Enforcement Profile – provisioned on Connectivity Plan creation
  - Microflow Profile – provisioned on Connectivity Plan creation
  - Network Context – provisioned on AS Routing Profile creation

The TWA configuration for EPCC provisioning is done at Operator level in `Operator` table of the TWA SQL database.
A TWA script `operator_epc_settings_update.sh` allows to configure easily the Operator, by setting in `Operator` table the following parameters:

- **epcProvisioningMode** – must be set to `HSS_SPR` when EPCC Connecter HSS is used or to `SPR` when external HSS is used
- **epcAddress** – must be set with the URL configured on HSS for the provisioning, i.e.: `http://192.168.242.120:18080` (IP Address of provisioning interface of HSS)
- **epcHssOperatorID** – must be set with the LTE Operator ID used, for ex.: EPC
- **epcHssKeyVersion** – must be set to AES128_keyvx where x must correspond to the key version provisioned on HSS to decrypt the Ki key (see section 13.2).
- **epcHssKey** – must be set to the secret key used to encrypt the K key. This secret key must be stored encrypted with TWA master private key.
- **epcHssApnID** – must be set to one of the APN id configured on HSS (see section 13.2)
  The default APN id is “apn”
- **epcLogin** – must be set to the login used to provision the HSS
  see section 13.2 on how to configure the login on HSS

- **epcPassword** – must be set to the password used to provision the HSS. see section 13.2 on how to configure the password on HSS
- **epcPGWAddress** – must be set with the URL configured on HSS for the provisioning of PGW objetcs (Fleets, Network Contexts, Enforcement and Microflow profiles), i.e.: `http://10.100.31.97:8888` (HSS floating IP address in case of HA)
- **epcPGWLogin** – must be set to the login used to provision the PGW objects
- **epcPGWPassword** – must be set to the password used to provision the PGW objects
- **epcPGWQoSProfile** – must be set to the QoS Profile name configured on PGW, i.e.: `qos1`

13.2 HSS/SPR Configuration

This section will explain all the necessary changes required on HSS/SPR module within EPC Connector.

13.2.1 Network Interface Configuration

By default, the HSS is configured with 3 different Network Interfaces with the following configuration:

- 1 Network Interface for Management
- 1 Network Interface for Diameter
- 1 Network Interface for SPR link with PGW

The configuration of Networks Interfaces is done through OS configuration file.

1. Edit the file `/etc/network/interfaces` file which contains the network configuration
2. Change `address`, `netmask` and `gateway` of `eth0` interface to be the one assigned for Management
3. Change Change `address` and `netmask` of `eth1` interface to be the one assigned for Diameter
4. Change `address` and `netmask` of `eth2` interface to be the one assigned for SPR link with PGW
5. Restart the network to apply the new network configuration by running the command `/etc/init.d/networking restart`

Below an example of the `interfaces` file after applying the changes

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
       address 192.168.242.120 → Management
       netmask 255.255.255.0
       gateway 192.168.242.254

allow-hotplug eth1
iface eth1 inet static
       address 192.168.247.120 →  Diameter
       netmask 255.255.255.0

allow-hotplug eth2
iface eth2 inet static
       address 192.168.243.120 → SPR Link with PGW
       netmask 255.255.255.0
```

13.2.2 HSS Provisioning Configuration

By default, the HSS/SPR is configured to listen on all network interfaces (port 18080) for the provisioning from TWA. In order that HSS listens only on one specific interface for secuirity reasons, the following configuration must be applied:

1. Edit the file `/home/actility/conf/tomcat-up.conf`
2. Set the parameter `instance[Default].http-binding-address` with the IP address where you want that HSS listens for provisioning. For example:

```
# Instance: Default
instance[Default].http-binding-address=192.168.242.120
instance[Default].http-binding-port=18080
instance[Default].https-binding-address=0.0.0.0
instance[Default].https-binding-port=18443
instance[Default].webapp-list=ROOT,device-provisioning-1.0
```

3. Restart tomcat by the running the command `monit restart tomcat-up`

The authentication mode and login/password must be configured in file `/home/actility/conf/device-provisioning.cfg`. The example configuration is shown below:

```
# ------------------------
# General configuration
# ------------------------

use-actility-hss:yes

# Log level: 0 -> No logs
#            1 -> Erros only
#            2 -> Errors, warnings and notices (recommended)
#            3 -> All (debug only)
log-level:2

# ------------------------
# Authentication
# ------------------------
```

```
# authentication can be : no , http-basic or http-digest
authentication:http-basic

# password-storage-mode can be : clear or crypted
password-storage-mode:crypted

# user list:
# use the following syntax :  user:<login>,<password>
# to add a user with a crypted password, type in the shell the following command:
#   add-crypted-user /home/actility/conf/device-provisioning.cfg <login> <clear-password>

user:actility,3789768c26280fb66d66018bac257d03
```

13.2.3 HSS Operator Table

The file `/var/actility/hss/scripts/table.OPTable.xml` allows to configure the different Operator key.

The Operator key must be stored encrypted by using the script `/usr/lib/actility/LP/HSS/cmd/crypt2.x`:

```
HSS01:/usr/lib/actility/LP/HSS/cmd# ./crypt2.x 0123456789ABCDEF0123456789ABCDEF
ef0532ba585a26f4e5b3ce4824fc4686
```

Here is an example of the Operator ID configured as `EPC` corresponding to the unencrypted Op Key `0123456789ABCDEF0123456789ABCDEF` (see below in red):

```
HSS01:/usr/lib/actility/LP/HSS/cmd# cat /var/actility/hss/scripts/table.OPTable.xml
<?xml version="1.0" encoding="UTF-8"?>
<CSM>
 <table name="OPTable">
   <entry key="EPC"  opkey="ef0532ba585a26f4e5b3ce4824fc4686"/>
   <entry key="Blue"  opkey="000102030405060708090A0B0C0D0E0F"/>
   <entry key="Actility"  opkey="050bfba4e9f600ad0c7f8095c1ad162d"
op="1006020f0a478bf6b699f15c062e42b3"/>
 </table>
</CSM>
```

13.2.4 HSS APN Table

The file `/var/actility/hss/scripts/table.HAPNTable.xml` allows to configure the different APN default settings.

The default configuration is:

```
<?xml version="1.0" encoding="UTF-8"?>
<CSM>
 <table name="HAPNTable">
   <entry key="apn"
               APNId="1"
               Context_Identifier="1"
               Service_Selection="default"
               PDN_Type="IPv4"
               QCI="5"
               ARP_Level="2"
               ARP_Capability="PRE_EMPTION_CAPABILITY_ENABLED"
               ARP_Vulnerability="PRE_EMPTION_VULNERABILITY_DISABLED"
               Charging_Characteristics="2048"
```

```
                MBR_DL="10000"
                MBR_UL="10000"
                RAT_Freq_Prio="256"
                PGWId="PGW1"
                NIDD_Mechanism="None"
                Preferred_Data_Mode="0"
   />
 </table>
</CSM>
```

13.2.5 Ki Key encryption - HSS Key and Key Version

The Ki key provisioned by TWA is encrypted with `EpcHssKey`. The `EpcHssKeyVersion` allows the HSS to select the right secret key to decrypt the Ki key. The secret keys are stored on HSS in file `/usr/lib/actility/LP/HSS/hss/keys.xml` and must be stored encrypted by using the following script:

```
HSS01:/usr/lib/actility/LP/HSS/cmd# ./crypt2.x 0123456789ABCEDF0123456789ABCDEF
ba8ca55c79695aa852d3406fb364637e
```

In the example below, the secret key number 1 corresponds to the `EpcHssKeyVersion` `AES128_keyv1` as an example configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<HSS>
 <keys>
   <key number="0" value="0123456789ABCEDF0123456789ABCDEF"/>
   <key number="1" value="ba8ca55c79695aa852d3406fb364637e"/>
   <key number="2" value="0123456789ABCEDF0123456789ABCDEF"/>
   <key number="3" value="0123456789ABCEDF0123456789ABCDEF"/>
   <key number="4" value="0123456789ABCEDF0123456789ABCDEF"/>
   <key number="5" value="0123456789ABCEDF0123456789ABCDEF"/>
 </keys>
</HSS>
```

13.2.6 HSS PLMN Table

The file `/var/actility/hss/scripts/table.VPLMNTable.xml` allows to configure the different PLMN allowed on HSS.

The example configuration is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<CSM>
 <table name="VPLMNTable">
   <entry key="PLMN00101"  allowed="1"/>
   <entry key="PLMN20893"  allowed="1"/>
   <entry key="PLMN50801"  allowed="1"/>
 </table>
</CSM>
```

13.2.7 Diameter and SPR configuration

Edit the file `/home/actility/conf/DRM.ini` and modify the parameters in red to match with your network configuration:

```
[DRM-starter]
       traceOn=stdout
```

```
       traceMaxSize=0
       traceLevel=0
       root=/usr/lib/actility/LP

############ HSS ############
[DRM-module]
      name=DSP-HSS
      dir=DSP/bin
      file=DSP.x -n DSP-HSS
      disable=0

[DSP-HSS-general]
      traceOn=/tmp/DSP-HSS.log
      traceMaxSize=0
      traceLevel=4
      tcpAddr=192.168.242.120 → HSS Listening IP address for Diameter
      tcpPort=3868
      sctpAddr=192.168.242.120 → HSS Listening IP address for Diameter
      sctpPort=3868
      name=DSP-HSS
      syslog=0

[DSP-HSS-peer]
      number=0
      name=DSP-HSS
      sendDestinationHost=0
      destinationHost=mme.epc.mnc093.mcc208.3gppnetwork.org
      destinationRealm=epc.mnc093.mcc208.3gppnetwork.org
      destinationProtocol=TCP
      destinationAddr=192.168.242.130 → MME Listening IP address for MME
      destinationPort=3868
      originHost=hss.epc.mnc093.mcc208.3gppnetwork.org
      originRealm=epc.mnc093.mcc208.3gppnetwork.org
      vendorId=3GPP
      productName=ACTILITY-HSS
      authApplicationId=16777251
      acctApplicationId=16777251
      supportedVendorId=3GPP
      started=0

############ SPR ############

[DRM-module]
      name=DSP-SPR
      dir=DSP/bin
      file=DSP.x -n DSP-SPR
      disable=0

[DSP-SPR-general]
      traceOn=/tmp/DSP-SPR.log
      traceMaxSize=0
      traceLevel=4
      tcpAddr=192.168.243.120 → HSS Listening IP address for SPR
      tcpPort=3878
      sctpAddr=192.168.243.120 → HSS Listening IP address for SPR
      sctpPort=3878
      name=DSP-SPR
      syslog=0

[DSP-SPR-peer]
      number=0
      name=DSP-SPR
      sendDestinationHost=0
      destinationHost=pgw.epc.mnc03.mcc525.3gppnetwork.org
      destinationRealm=epc.mnc03.mcc525.3gppnetwork.org
      destinationProtocol=TCP
      destinationAddr=192.168.243.100 → PGW Listening IP address for SPR
```

```
        destinationPort=3878
        originHost=SPR.SPR.actility.com
        originRealm=SPR.actility.com
        vendorId=3GPP
        productName=ACTILITY-SPR
        authApplicationId=16777310
        acctApplicationId=16777310
        supportedVendorId=3GPP
        started=1
```

13.3 MTC-LTE-GW Configuration

13.3.1 Network Interface Configuration

On MTC-LTE-GW the network configuration is done as follows:

- As OS level for management IP address
- Through PGW shell for GTP and GRE (kernel level)

13.3.2 NW Interface configuration at OS level

The configuration of networks Interfaces is done through OS configuration file.

1. Edit the file `/etc/network/interfaces` file for applying the network configuration
2. Change `address`, `netmask` and `gateway` of `eth0` interface to be the one assigned for Management
3. Change `address` and `netmask` of `eth1` interface to be the one assigned for Message to LRC
4. Change `address` and `netmask` of `eth2` interface to be the one assigned for SPR link
5. Restart the network to apply the new network configuration by running the command `/etc/init.d/networking restart`

Below an example of the `interfaces` file after applying the changes:

```
PGW01:~/update# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
       address 192.168.242.100 → Management
       netmask 255.255.255.0
       gateway 192.168.242.254

allow-hotplug eth1
iface eth1 inet static
       address 192.168.247.100 → Message mode to LRC
       netmask 255.255.255.0
```

```
allow-hotplug eth2
iface eth2 inet static
       address 192.168.243.100 → SPR Link
       netmask 255.255.255.0

# eth3 and eth4 configured in PGW application in kernel level
```

13.3.3 NW Interface configuration through PGW shell (kernel level)

First you have at ESX level to map:

- eth3 interface to the network assigned for GTP link (corresponding to port A1 on PGW shell)
- eth4 interface to the network assigned for GRE link (corresponding to port A2 on PGW shell)

Then connect on the PGW shell and run the following command:

```
PGW01:~# shell
Local COSS controller version is 2.8.5
Usage : Type <tab> or help for commands and completion.

PGW01>
```

1. Change the IP address (GTP link with SGW) of the different aliases (ip name) used for GTP by the running the following commands:

```
PGW01> vdip ipname change GTPListener 192.168.247.101/24 context 0
```

Check that file /var/actility/pgw/config/service_parameters.xml has been modified accordingly, if it is not the case change the IP address manually in the file.

2. Delete the current GTP link by running the following command:

```
PGW01> vdip link del T1
```

3. Create the new GTP link with the right PGW IP address by running the following command:

```
PGW01> vdip link add T1 type gtp local GTPListener from context 0
```

4. Change the vdip address for PGW GTP link by running the following commands:

```
PGW01> vdip addr del 192.168.2.11/24 dev A1 context 0
PGW01> vdip addr add GTPListener dev A1 context 0
```

5. Change the vdip address for PGW GRE link by running the following commands:

```
PGW01> vdip addr del 192.168.1.200/24 dev A2 context 0
PGW01> vdip addr add 192.168.243.101/24 dev A2 context 0
```

   Check that file /var/vedicis/pgw/config/service_parameters.xml has been modified accordingly. If it is not the case change the IP address manually in the file.

6. Change the vdip address for PGW GRE link by running the following commands:

```
PGW01> vdip addr del 192.168.1.200/24 dev A2 context 0
PGW01> vdip addr add GREListener dev A2 context 0
```

Note: In case of HA (HSS cluster) a VIP address must be defined for PGW GRE link, in such case change it as follow:

```
PGW01> vdip addr virtual del 192.168.2.11/24 dev A2 context 0
PGW01> vdip addr virtual add GREListener GRELis dev A2 context 0
```

13.3.4 SPR Configuration

Edit the file `/home/actility/conf/DRM.ini` and modify the parameters in red to match with your network configuration:

```
[DRM-starter]
       traceOn=stdout
       traceMaxSize=0
       traceLevel=0
       root=/usr/lib/actility/LP

[DRM-module]
      name=DSP-SPR
      dir=DSP/bin
      file=DSP.x -n DSP-SPR
      disable=0

[DSP-SPR-general]
      traceOn=/tmp/DSP-SPR.log
      traceMaxSize=0
      traceLevel=2
      tcpAddr=192.168.243.100 → PGW Listening IP address for SPR
      tcpPort=3878
      sctpAddr=192.168.243.100 → PGW Listening IP address for SPR
      sctpPort=3878
      name=DSP-SPR
      syslog=0

[DSP-SPR-peer]
      number=0
      name=DSP-SPR
      sendDestinationHost=0
      destinationHost=SPR.SPR.actility.com
      destinationRealm=SPR.actility.com
      destinationProtocol=TCP
      destinationAddr=192.168.243.120 → HSS Listening IP address for SPR
      destinationPort=3878
      originHost=pgw.epc.mnc03.mcc525.3gppnetwork.org
      originRealm=epc.mnc03.mcc525.3gppnetwork.org
      vendorId=3GPP
      productName=ACTILITY-SPR
      authApplicationId=16777310
      acctApplicationId=16777310
      supportedVendorId=3GPP
      started=1
```

13.3.5 LRC Configuration for Message mode

Edit the file `/home/actility/conf/mtc.ini` and replace the parameter in red with the IP address of LRC:

```
[MTC-general]
       traceOn=/var/log/actility/mtc.log
       traceMaxSize=0
       traceLevel=0
       Device=R2
       #IPAddress=10.101.0.1
```

```
        #IPAddress=10.100.31.96
        LRRId=1
        UUID=imei
        Protocol=0
        PortUL=5000
        PortDL=5000
        TestMode=0

[MTC-LRC]
        number=0
        addr_1=192.168.247.9 → LRC IP address for Message mode
        port_1=2406
        #addr_2=192.168.1.11
        #port_2=1235
```

13.3.6 APN Routing Context Configuration

The file /var/actility/pgw/scripts/table.APNTable.xml allows to configure per APN:

- The associated Routing Context
- the subnet associated to the device (used to create the right route at kernel level)
- the DHCP Pool (IP address assigned to the device)
- DNS IP address provided to the device
- The QoS Profile

Below the current configuration where Default APN is associated to Routing Context id 99:

```
PGW01:~# cat /var/actility/pgw/scripts/table.APNTable.xml
<?xml version="1.0" encoding="UTF-8"?>
<CSM>
        <table name='APNTable'>
                <entry key='Default' id='99' subnet='192.168.3.192/26' qos='apnprofile1'
rc='99' dns='8.8.8.8' pools='192.168.3.200-192.168.3.250' process='1'/>
        </table>
</CSM>
```

13.4 Operator Manager/Device Profile

The first step to provisioning a Cellular LTE device is to create a “Cellular” profile in operator manager. The cellular profile contains generic information on cellular device capabilities.

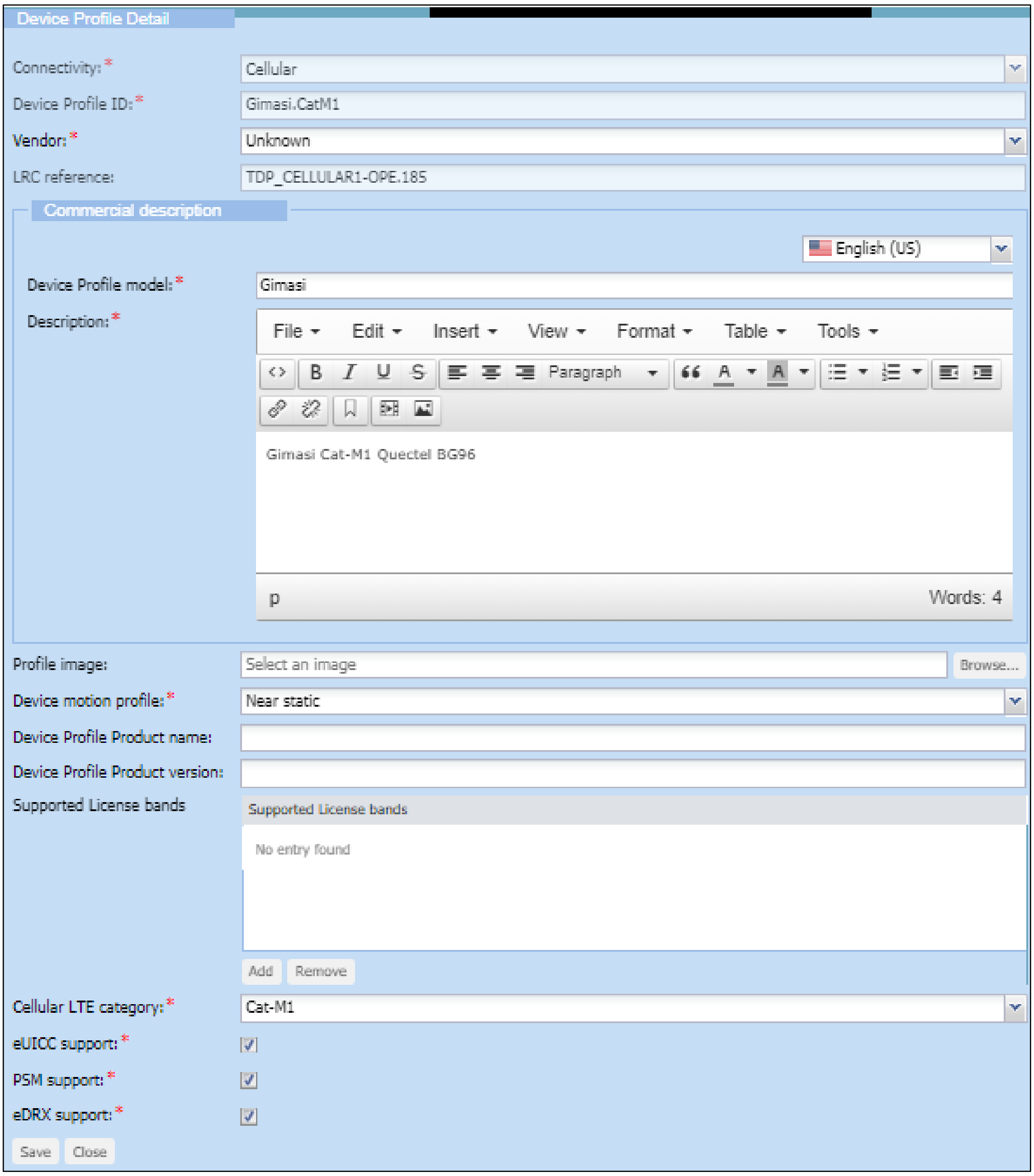


Figure 15 : Device Profile Creation

13.5 Operator Manager / Connectivity Supplier /Connectivity Plan

The figure below shows an example connectivity plan that applies to “message mode” and “Direct IP” traffic. It can only be created by connectivity supplier account that exists within operator manager.

Cellular Connectivity plan is divided into two parts:

1. **Message mode**: The connectivity plan parameters of this mode control all the message mode traffic. This mode is applicable to all the messages
2. **Direct IP mode**: The connectivity plan parameters of this mode control all the message mode traffic. These parameters are applicable to traffic that is routed directly from MTC-LTE-GW to Application servers bypassing ThingPark Wireless.

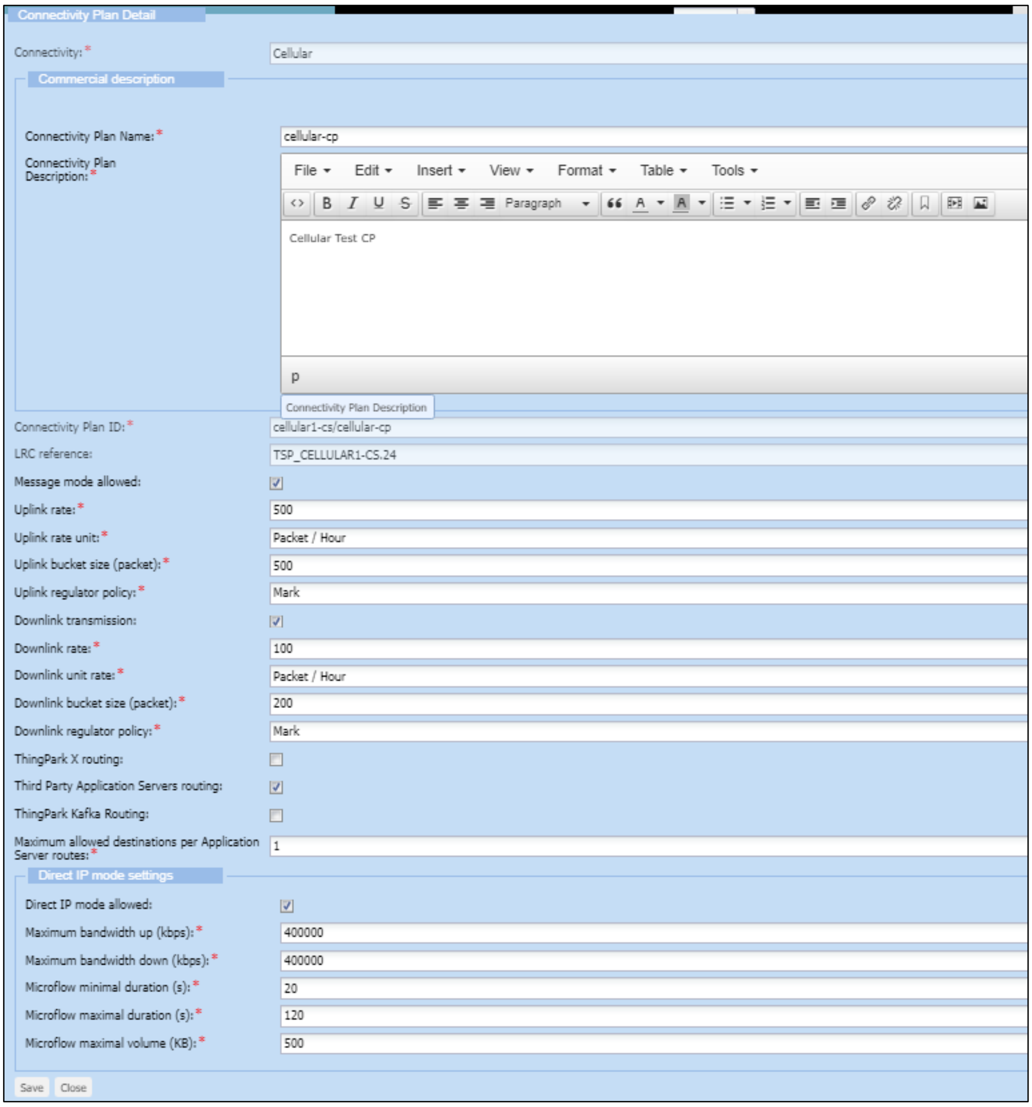


Figure 16 : Connectivity Plan Creation

13.6 Application Server

The next step is to create application server from device manager interface. In the example shown in figure below, the Application server IP address is listed in the destination field. Note, that this application server is only for handling "message mode" traffic.

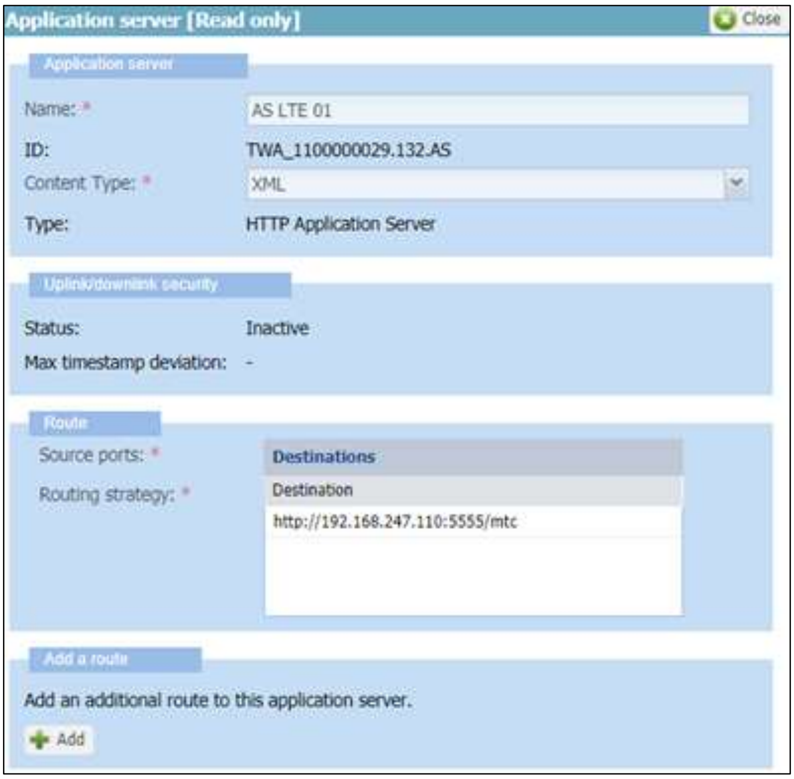


Figure 17 : Application Server Configuration

13.7 AS Routing Profile

After that, Application Server (AS) Routing profile is created to link the Application server to it. This will allow LRC to route “message mode” or “Direct IP” traffic for the devices that belong to a particular IoT subscriber to the Application Server (AS) listed in earlier section. AS Routing profile also has configuration for UDP source port (for ex. 7777) which is used to identify message mode packets.

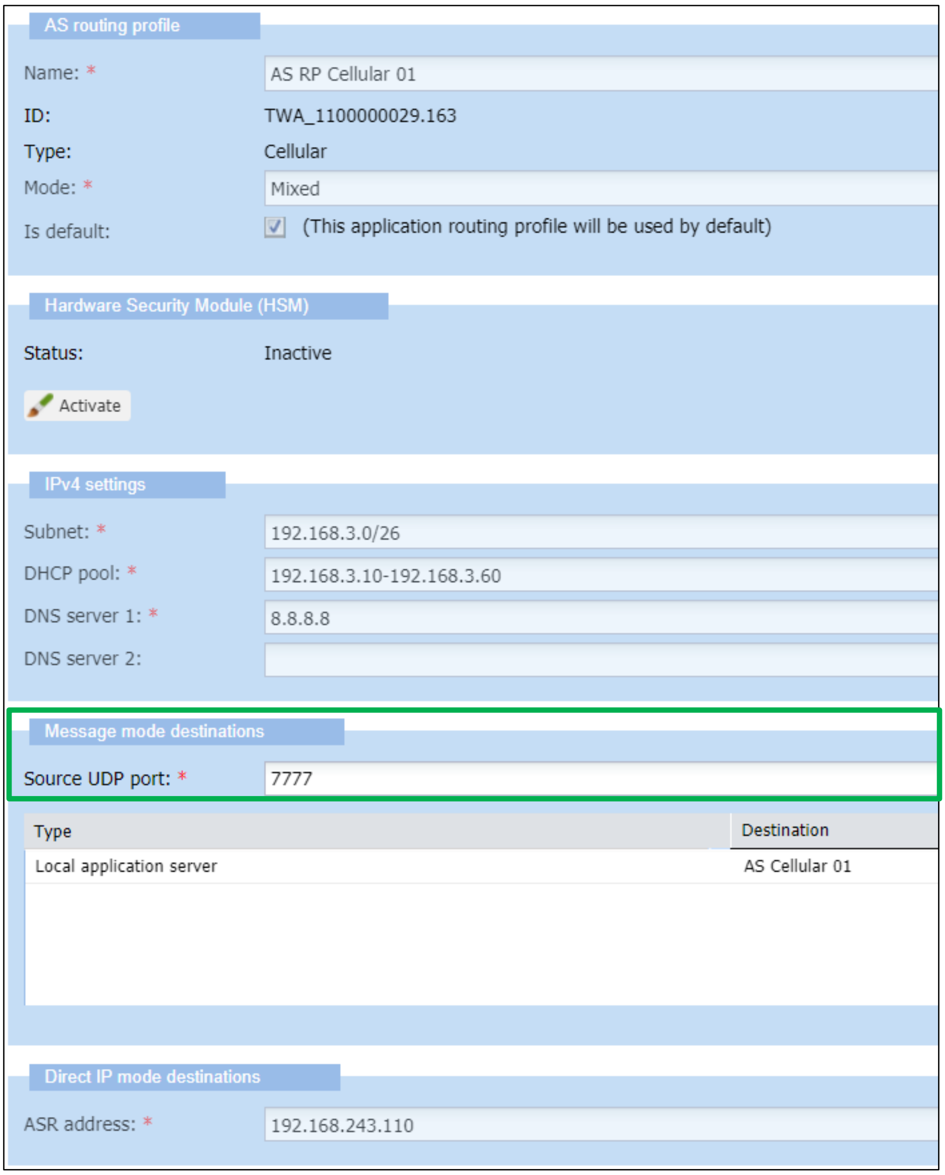


Figure 18 : Routing Profile Configuration

13.8 Device manager: Creation of devices

Once the device profile is created in operator manager, IoT subscriber account can add devices from the device manager. Here are the following fields that must be carefully entered in device manager for cellular device to connect successfully to the cellular network, EPC Connector and ThingPark Wireless.

- **IMEI**: This is the identity of the module and is printed on the module. The IMEI on the module must match to that provisioned in ThingPark Wireless

- **IMSI**: This is the identifier that is present inside the SIM card and must match to that provisioned in ThingPark Wireless
- **Ki**: This is the device secret Key that is present inside the SIM card and must match to that provisioned in ThingPark Wireless

The new device created should be mapped to the connectivity plan and routing profile that were created in earlier sections.

Figure 19 : New Device Creation

13.9 Device Manager: Statistics

The figure below shows the device manager statistics once the device is registered on the cellular network. Here are the important parameters that are shown in the statistics:

- **IMEI**: It is the IMEI that was provided during device creation
- **IP Address**: It is the IP Address of the device that is provided from the core network. It is listed here for information purposes

- **Serving network MCC/MNC**: It is the MCC/MNC that is provided from SGW to the EPC Connector
- **Uplink/Downlink packets**: This is the statistics of message mode traffic that goes through LRC for this device
- **Direct IP mode bytes**: This is the statistics of the direct IP based microflow reports

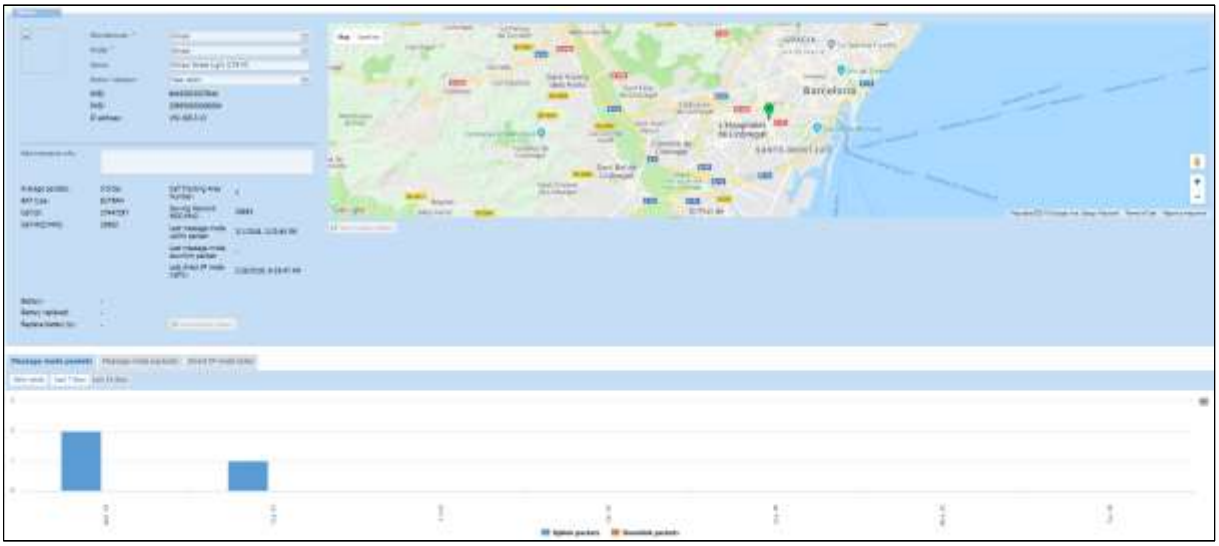


Figure 20 : Device Manager Statistics

13.10 Device Manager: Listing of devices

The figure below shows different LoRaWAN™ and 3GPP based IoT devices within the same IoT subscriber. The columns of the list give you the following information on the displayed devices:

- **Name/Type**: name and device profile
- **Identifier**:
  - **For LoRaWAN™™ devices**: **DevEUI** and **DevAddr** of the device: similar to a MAC address and a Network address
  - **For cellular devices**: **IMEI** and **IP** address are used for identification
- **Connectivity**: Connectivity plan and AS routing profile
- **Average packets**: number of packets/day (not applicable to cellular devices)
- **Mean PER**: mean Packet Error Rate (not applicable to cellular devices)
- **Average SNR**: based on the last five packets received (Not applicable to cellular devices)
- **Battery** (not applicable to cellular devices)
- **Alarm**: number of alarms not acknowledged
- **Locate**: opens the Device location window displaying the device on a map

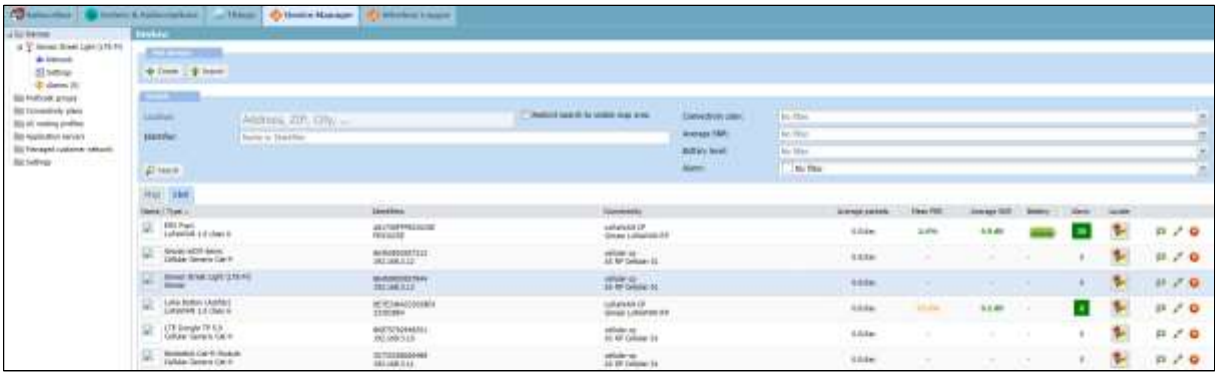


Figure 21 : Device Manager listing of devices

13.11 Wireless logger

Wireless logger is very useful tool within ThingPark Wireless information to track the "message mode" traffic that goes through LRC between the device and Application server (AS). It also shows the microflow reports that represent "Direct IP" traffic passing through MTC-LTE-GW directly to Application Servers. Once the device is provisioned in ThingPark Wireless, it has five distinct types of packets in wireless logger specifc to LTE:

1. **Session Creation**: This is an uplink message in wireless logger when the device successfully connects to the eNodeB/Core network and is successfully registered to the network. It is created in the last step of LTE Attach procedure when SGW of the operator's core network sends session creation request to the MTC-LTE-GW within EPC Connector, which then forwards this to LRC within ThingPark Wireless. The snapshot of session creation packet is shown below.

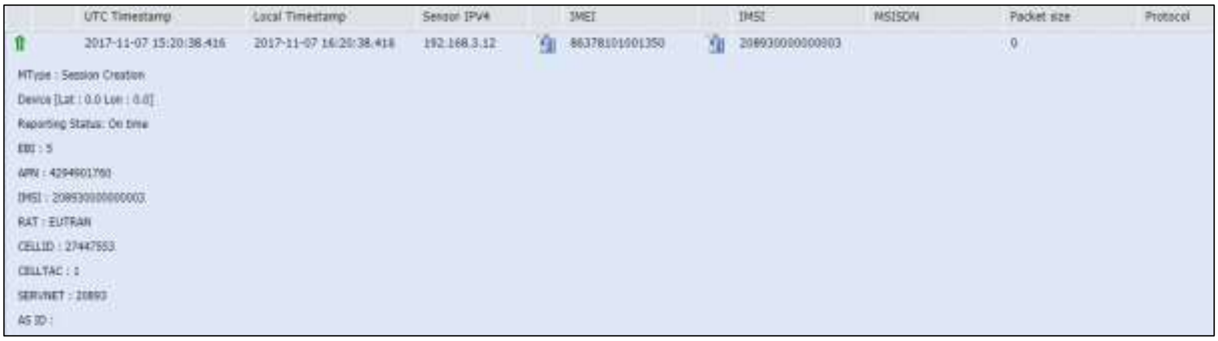


Figure 22 : Wireless logger (session creation)

2. **Uplink Payload**: This is the typical uplink UDP payload that is captured by LRC when the device is doing uplink "message mode" traffic between itself and the Application Server (AS). Note, that the UDP source port is 7777 which classifies uplink UDP packet received by MTC-LTE-GW as a "message mode" packet. This port is configurable in the AS Routing profile.

Figure 23 : Wireless logger (uplink payload)

3. **Downlink Payload**: This is the typical downlink payload that is captured by LRC when the AS is doing downlink “message mode” traffic between itself and the device. However, the device needs to always initiate uplink traffic first with the AS before AS can start any downlink traffic.

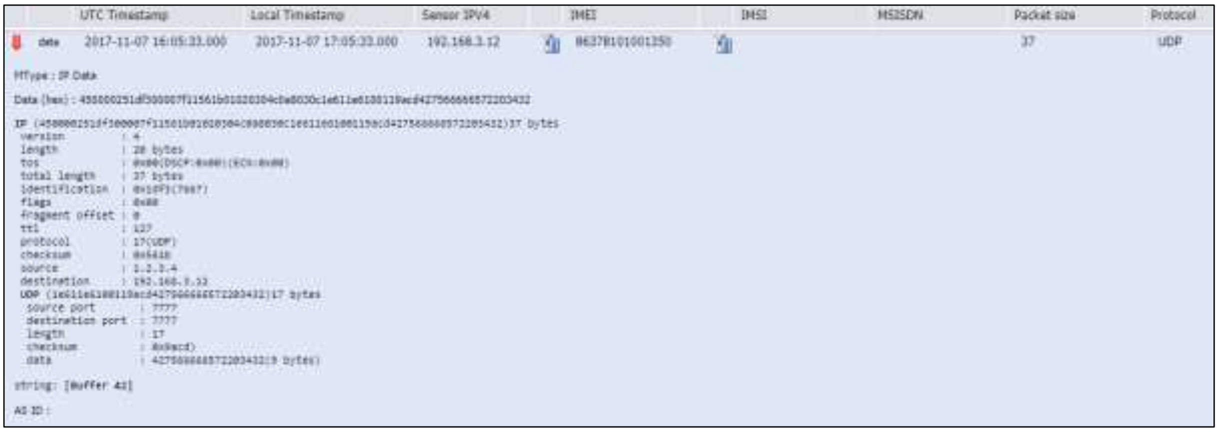


Figure 24 : Wireless logger (downlink payload)

4. **Session Deletion**: This message is received when the device detaches from the network for some reason (for example, when it loses connectivity or is switched off). This message is sent from MTC-LTE-GW to LRC upon receiving of session deletion request from SGW.

Figure 25 : Wireless logger (session deletion)

5. **Microflow Report**: This message is sent from MTC-LTE-GW to LRC whenever there is Direct IP traffic. It is not the actual payload but only the statistics of the traffic that passes though MTC-LTE-GW.

Figure 26 : Wireless logger (Microflow Reports)

14 UNSUPPORTED FEATURES

Following features are not supported in EPC Connector:

- 2G/3G device connections
- Voice Call features

15 EPC CONNECTOR NEXT RELEASES ENHANCEMENTS

- Support of SCEF/NIDD over T6a

About Actility

Actility is an industry leader in LPWAN (Low Power Wide Area) large scale infrastructure with ThingPark™, the new generation standard-based IoT/M2M communication platform. Actility's ThingPark Wireless™ network provides long-range coverage for low-power sensors used in SmartCity, SmartBuilding and SmartFactory applications. Actility also provides the ThingPark X which provides big data storage for sensor data and exposes sensor function through an open API allowing developers to provide vertical applications on top of rolled out sensors. To help vendors transform their sensors, Actility provides the ThingPark IoT platform which include embedded software solutions and cloud solutions to help devices connect to innovative applications. Via the ThingPark Market, an online marketplace engine dedicated to the IoT sensors, applications and network solutions, Actility enables the roll-out of new innovative IoT services for sensor vendors and network solution vendors. Actility is a founding member of the LoRa Alliance™: the largest, most powerful standards-based effort to enable the Internet of Things (IoT). Visit www.actility.com.