As main summary, we permit only the publishment of messages that you have defined for the BDM toll referenced into the "ENACT\_DevOps\_Apps\_Interfaces - Context Monitoring v0.3" specification for the ENACT tool:

**-J100106. Wagon position WSN solution JSON payload**

The payload produced by the coordinator of the Wagon position based on the WSN is presented. This packet presents the triple of data required to calculate the Train Integrity, each packet carries the information of every node of the composition.

...Three resources in this order:

**Accelerometer (3313): [m/s^2]**

**GNSS (3336):[ ]**

**RSSI (3300:generic sensor): [dBm]**

**-J100108. Train Integrity Report JSON payload**

These payloads will leave the CMW from the Business Layer. The packet informs about the general state of the composition Train Integrity. The broker must publish messages that has a switch sensor (SensorID: 3300).

The values for the generic sensor are listed following the updated SubSet-026: **[ ]**

**0: No Train Integrity information**

**1: Train Integrity confirmed by the SIP**

**2: Train Integrity confirmed by the Driver**

**3: Train Integrity lost**

**-J102119. Energy WSN JSON payload**

This packets contents the data reported by the Maintenance WSN. The broker must register messages that has a Wagon Plate (SensorID: 3341) following the European Guideline for the Identification of Railway Assets using GS1 Standards [6],

**Train ID (SensorID: 3341) identifies the Train ID that represents the composition. Eg:101101 [ ]**

**Watts sensor (SensorID: 3305): [W]**

**Voltage sensor (SensorID: 3316): [V]**

**-J102120. On Track Logistics JSON payload**

This packets contents the data reported by the On Track Logistics WSN. The broker must register messages that have a Wagon Plate (SensorID: 3341) following the European Guideline for the Identification of Railway Assets using GS1 Standards [6],

**Train ID (SensorID: 3341) identifies the Train ID that represents the composition. Eg:101101 [ ]**

**Logical Position (3300:generic sensor): [ ]**

## Information we have in the simulator :

* speed : km/h (**Not used into the Use Case**)
* power : m/s² (**those units?**)
* acceleration : m/s² (**Ok for the Use Case**)
* relative distance to the starting train station : m (**Not used into the Use Case**).
  + RSSI is distance **between wagons** (distance between wagon 1 and 2, distance beteween the wagon 2 to 3,….). There is nothing related with something external to the train into the Use Case-
* relative distance to the others train station : m **(Not used into the Use Case).**
  + RSSI is distance **between wagons**. There is nothing related with something external to the train into the Use Case-
* relative distance to the next train : m **(Not used into the Use Case).**
  + RSSI is distance **between wagons**. There is nothing related with something external to the train into the Use Case-
* elevation : m **(GPS?)**
* temperature: °C (**Not used into the Use Case).**
* air pressure : kPa **(Not used into the Use Case).**
* air density : kg/m3 **(Not used into the Use Case).**
* total mass : kg **(Not used into the Use Case).**

## Needed information:

* Units Indra uses for sensors: Temperature (for instance: K, C°, F°), Load, Air pressure, Acceleration **(responded before)**
* Distance Sensor Object definition : what distance should this object measure (relative to the starting train station, next train station, next train,...) ? **(responded before)**
* What is the method to generate CRC ? A procedure is described in the document but we do not have access to the algorithm that generate the CRC **(We have provided the data sections that must be included into the CRC calculus for each field, the CRC method is the standard CRC32, note: can be phyton libraries)**
* What source/subsource ID should we use ? **For these usage, as EDI is publishing also these messages, the Source must be equal to the EDI one (121). The Subrouce is a parameters associated to each publisher (each train). Therefore, if you have 100 publishers you require 100 different subsources. As they must be different to the ones used by EDI, start from the 200 increasing one unit per publisher (e.g: the first publisher would be Source: 121, Subsource:200 and a second publisher would be Source: 121, Subsource:201,…)**
* Some details about services and subservices so we can know what we need to use.

Here I can respond to some of your fields.

**-"Service ID":** 100106 (in case of you publish the message 100106, 100108 in case you publish the 100108 message, …)

**-"Gateway":** 0

**-"Source":** 0

**-"Node ID":** (for the publisher with a Source: 121 and Subsource: 200):121200XXXXXXXXX (XXXXXXXXX is a number to chosen by the partner. Eg: Publisher 1: 121200000000000 and Publisher 2: 121201000000000)

**-"Timeaccuracy":** (As it is mentioned in the "ENACT Canonical Data Model Specification v1.7 document", the "**TimeAccuracy:** This field represents a field that establish the accuracy of the Timestamp. It only contents the tenths of seconds to nanoseconds part.")

Basically, it establishes the time resolution until nanoseconds. The Timestamp and the Timeaccuracy are appended and the complete time mark is completed.)

Eg:

"TimeStamp": 1592211934, ->readable date: 24/09/2020-9:32:30

"TimeAccuracy": 396395480,

Complete timemark: 1592211934, 396395480 -> readable date: 24/09/2020-9:32:30 and 396395480 nanoseconds.

## Example of json we can generate currently :

{

"Service ID": "Integer",

"Root": [

{

"Gateway": "Integer",

"Source": "Integer",

"TimseStamp": 1600264370848

}

],

"Nodes": [

{

"Safety": true,

"NodeID": "Integer",

"TimeStamp": 1600264370847,

"TimeAccuracy": "Integer",

"Sensors-Actuators": [

{

"SensorID": 3313,

"TimeStamp": 1600264370492,

"TimeAccuracy": "Integer",

"Resources": {

"5701": "m/s²",

"5702": 0.649010021971197,

"5703": 0.00437392851420952,

"5704": -0.0028816290770528902

}

}

],

"CRC": "Integer"

},

{

"Safety": true,

"NodeID": "Integer",

"TimeStamp": 1600264370847,

"TimeAccuracy": "Integer",

"Sensors-Actuators": [

{

"SensorID": 3303,

"TimeStamp": 1600264369945,

"TimeAccuracy": "Integer",

"Resources": {

"5601": 19.9821457923245,

"5602": 19.98224857645,

"5700": 19.9822451105845,

"5701": "Cel"

}

},

{

"SensorID": 3323,

"TimeStamp": 1600264370490,

"TimeAccuracy": "Integer",

"Resources": {

"5700": 101325.0076,

"5701": "kPa"

}

},

{

"SensorID": 3322,

"TimeStamp": 1600264370607,

"TimeAccuracy": "Integer",

"Resources": {

"5601": 101325.0076,

"5602": 101325.0076,

"5700": 193290,

"5701": "kg"

}

}

],

"CRC": "Integer"

}

],

"CRC": "Integer"

}