FIWARE-EPCIS MEDIATION GATEWAY DEVELOPER GUIDE

Developer Guide_V_1.0.0

Abstract

This document includes how to use FIWARE-EPCIS mediation gateway. This document doesn't include how to use EPCIS and FIWARE in detail. It assumes the user is familiar with each of them fairly.

Yalew kidane

yalewkidane@gmail.com/@kaist.ac.kr

Contents

Servers to be prepared

FIWARE server

Use the following page to install FIWARE

https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals/admin/yum.md

EPCIS server

- Option 1:
  - Make sure you have installed mongodb/mysql whichever you are using
  - Download epcis war file from here: https://github.com/yalewkidane/FIWARE_EPCIS_Mediation_Gateway/tree/master/target
  - Download the apache tomcat 8 from here https://tomcat.apache.org
  - After extracting the apache tomcat file put the epcis war file in to ***path/to/your_tomcat_download/apache-tomact-8.x.xx/ webapps***
  - [for Linux] On terminal go ***path/to/your_tomcat_download/apache-tomact-8.x.xx/bin/***
  - [for Linux] sh ./catalina.sh run
  - [for Window] use .bat file
- Option 2:
  - Follow the instruction the original EPCIS GitHub : https://github.com/JaewookByun/epcis

Mediation gateway server

Status of FIWARE to EPCIS mediation gateway

- Current implementation only include test models (Car and Room) and Pig farming
- In the near future a more general translation module will be introduced
- Anyone interested in other data models can follow the same principle used in farm model implementation to extend to other domains

Version

FIWARE_EPCIS_MediationGateway_V_1.0.0

Source code

https://github.com/yalewkidane/FIWARE_EPCIS_Mediation_Gateway

Jar file

https://github.com/yalewkidane/FIWARE_EPCIS_Mediation_Gateway/blob/master/target/fiware_oiliot_mediation-0.0.1-SNAPSHOT.jar

How to run the mediation gateway

From https://github.com/yalewkidane/FIWARE_EPCIS_Mediation_Gateway/blob/master/target/ download fiware_oiliot_mediation-0.0.1-SNAPSHOT.jar

Run with the following command:

java -jar path/to/your_jar_file/ fiware_oiliot_mediation-0.0.1-SNAPSHOT.jar

Enter FIWARE server URL (e.g localhost:2016) :

localhost:1026

Enter FIWARE server URL (e.g localhost:8080) :

localhost:8080

Mediation Gateway Port (e.g 8083):

8081

After that, the mediation gateway will run and you can access the interface through any browser

localhost: Mediation_Gateway_Port /home

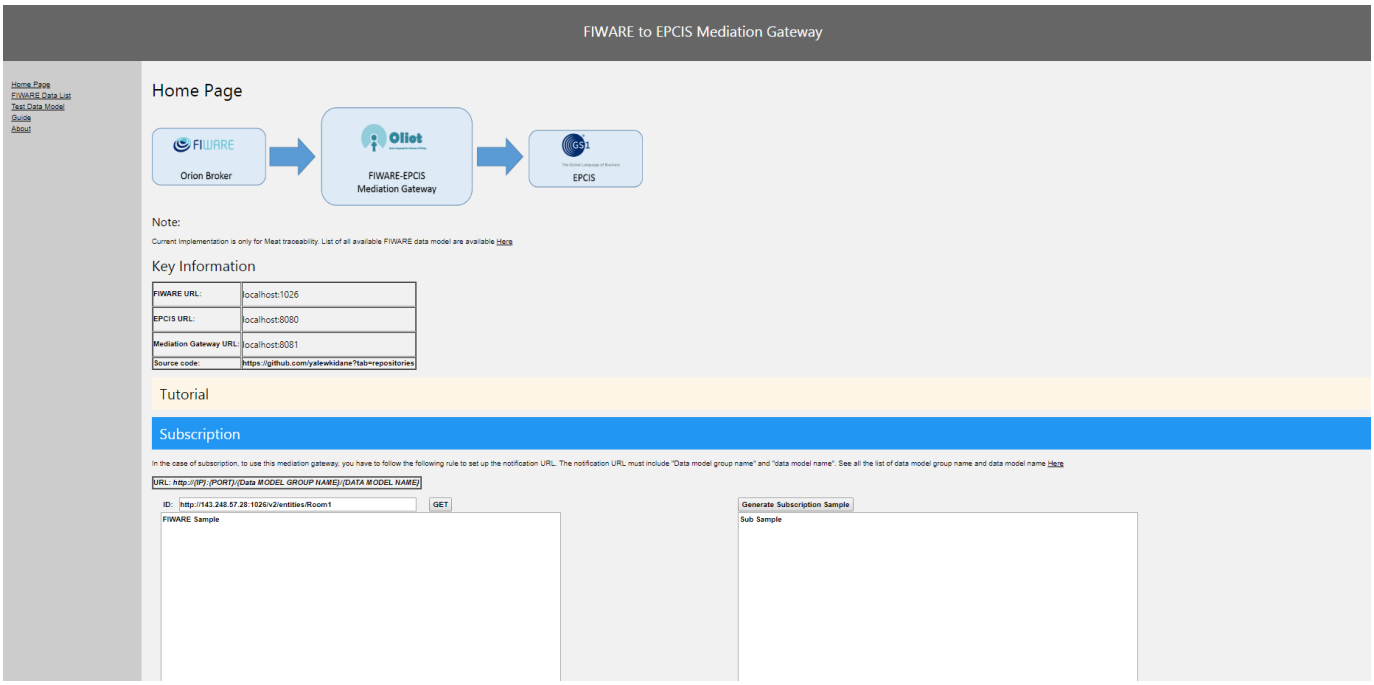
From the above example the url should be → localhost:8081/home

After that you will see the interfaces presented below

FIWARE to EPCIS mediation Gateway Interfaces

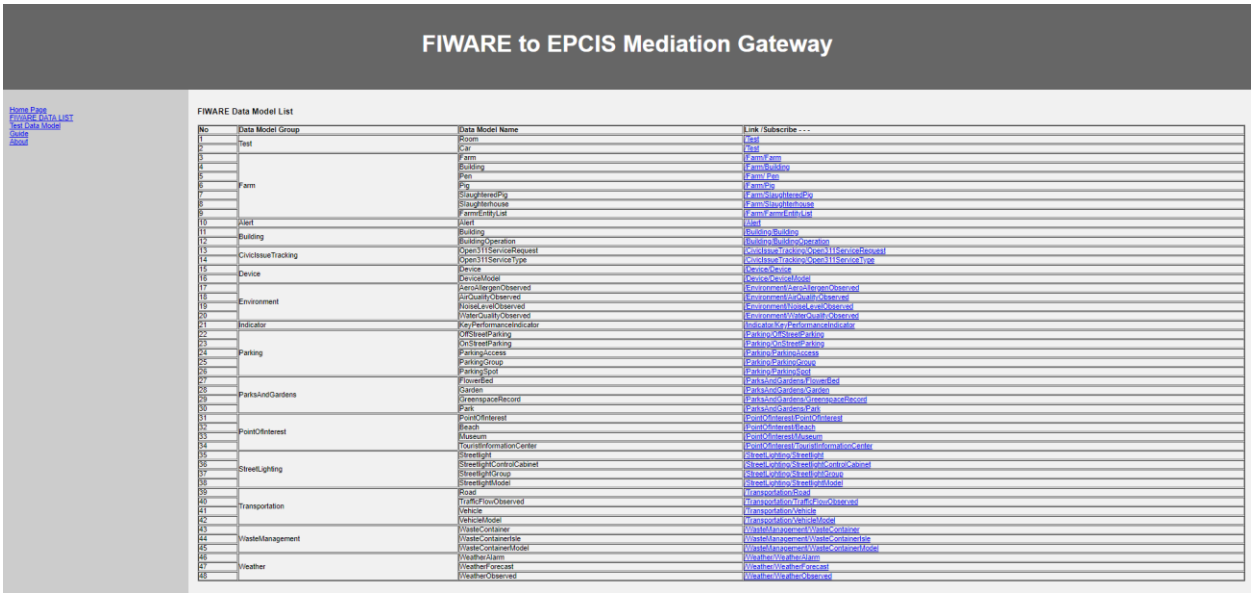
Main page

{IP}:{PORT} /home



Fiware Data List

{IP}:{PORT} /FiwareDataModel



FIWARE to EPCIS Mediation Gateway

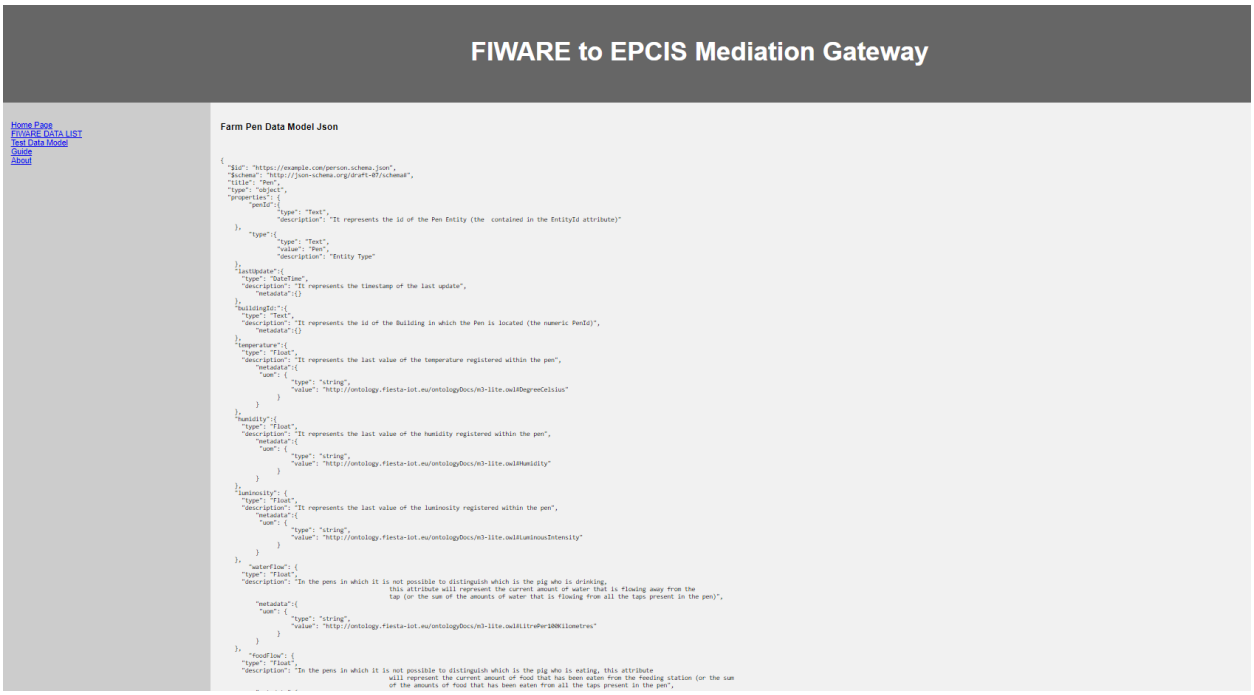
Home Page
FIWARE DATA LIST
Test Data Model
Guide
About

FIWARE Data Model List

No	Data Model Group	Data Model Name	Link / Subscribe - - -
1	Test	Room	/Test
2		Car	/Test
3	Farm	/Farm	/Farm/Farm
4		/Building	/Farm/Building
5		/Pen	/Farm/Pen
6		/Pig	/Farm/Pig
7		/SlaughteredPig	/Farm/SlaughteredPig
8		/Slaughterhouse	/Farm/Slaughterhouse
9		/FarmEntityList	/Farm/FarmsEntityList
10	Alert	/Alert	/Alert
11	Building	/Building	/Building/Building
12		/BuildingOperation	/Building/BuildingOperation
13	CivicIssueTracking	/Open311ServiceRequest	/CivicIssueTracking/Open311ServiceRequest
14		/Open311ServiceType	/CivicIssueTracking/Open311ServiceType
15	Device	/Device	/Device/Device
16		/DeviceModel	/Device/DeviceModel
17	Environment	/AeroAllergenObserved	/Environment/AeroAllergenObserved
18		/AirQualityObserved	/Environment/AirQualityObserved
19		/NoiseLevelObserved	/Environment/NoiseLevelObserved
20		/WaterQualityObserved	/Environment/WaterQualityObserved
21	Indicator	/KeyPerformanceIndicator	/Indicator/KeyPerformanceIndicator
22	Parking	/OffStreetParking	/Parking/OffStreetParking
23		/OnStreetParking	/Parking/OnStreetParking
24		/ParkingAccess	/Parking/ParkingAccess
25		/ParkingGroup	/Parking/ParkingGroup
26		/ParkingSpot	/Parking/ParkingSpot
27	ParksAndGardens	/FlowerBed	/ParksAndGardens/FlowerBed
28		/Garden	/ParksAndGardens/Garden
29		/GreenspaceRecord	/ParksAndGardens/GreenspaceRecord
30		/Park	/ParksAndGardens/Park
31	PointOfInterest	/PointOfInterest	/PointOfInterest/PointOfInterest
32		/Beach	/PointOfInterest/Beach
33		/Museum	/PointOfInterest/Museum
34		/TouristInformationCenter	/PointOfInterest/TouristInformationCenter
35	StreetLighting	/Streetlight	/StreetLighting/Streetlight
36		/StreetlightControlCabinet	/StreetLighting/StreetlightControlCabinet
37		/StreetlightGroup	/StreetLighting/StreetlightGroup
38		/StreetlightModel	/StreetLighting/StreetlightModel
39	Transportation	/Road	/Transportation/Road
40		/TrafficFlowObserved	/Transportation/TrafficFlowObserved
41		/Vehicle	/Transportation/Vehicle
42		/VehicleModel	/Transportation/VehicleModel
43	WasteManagement	/WasteContainer	/WasteManagement/WasteContainer
44		/WasteContainerIsle	/WasteManagement/WasteContainerIsle
45		/WasteContainerModel	/WasteManagement/WasteContainerModel
46	Weather	/WeatherAlarm	/Weather/WeatherAlarm
47		/WeatherForecast	/Weather/WeatherForecast
48		/WeatherObserved	/Weather/WeatherObserved

Farm Data List example (Schema and example)

{IP}:{PORT} / farm/pen



More example:

- {IP}:{PORT} / farm/farm
- {IP}:{PORT} / farm/building
- {IP}:{PORT} / farm/pig
- {IP}:{PORT} / farm/slaughteredPig
- {IP}:{PORT} / farm/slaougterhouse
- {IP}:{PORT} / farm/entityList

Simple Subscription example

1. Check for a specific entities on FIWARE before subscription (eg. Room8)

Server	[FIWARE] localhost:1026/v2
Method	GET
URL	localhost:1026/v2/entities/Room8
Headers	Content-Type: application/json
Status	404 Not Found

Response	{
	"error": "NotFound",
	"description": "The requested entity has not been found. Check type and id"
	}

Comment	Entity Room8 doesn't exist in FIWARE so we need to create it first

2. Add a Room entity to FIWARE before subscription (eg. Room8)

Server	[FIWARE] localhost:1026/v2
Method	POST
URL	localhost:1026/v2/entities
Headers	Content-Type: application/json

Body	{
	"id": "Room8",
	"type": "Room",
	"pressure": {
	"type": "Integer",
	"value": 123,
	"metadata": {}
	},
	"temperature": {
	"type": "Float",
	"value": 28,
	"metadata": {}
	}
	}

Status	201 Created
Response	{}
Comment	Entity Room8 is created

3. Check the created entity

Server	[FIWARE] localhost:1026/v2
Method	GET
URL	localhost:1026/v2/entities/Room8
Headers	Content-Type: application/json
Status	200 OK

Response	{
	"id": "Room8",
	"type": "Room",
	"pressure": {
	"type": "Integer",
	"value": 123,
	"metadata": {}
	},
	"temperature": {
	"type": "Float",
	"value": 28,
	"metadata": {}
	}
	}

Comment	Entity Room8 created in step 2 is returned

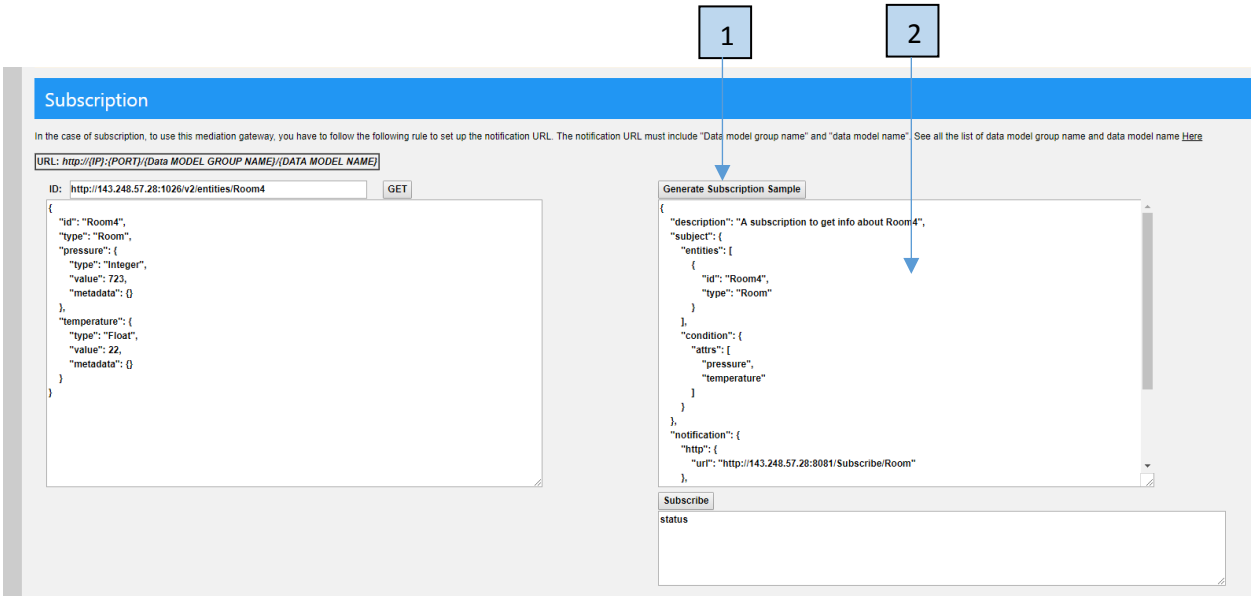
4. Generate sample subscription.

How to make subscription body	Refer:- https://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html
How to make subscription URL	To use this mediation gateway, you have to follow the following rule to set up the notification URL. The notification URL must include "Data model group name" and "data model name" and it should looks like this *http://{IP}:{PORT}/Subscribe/{DATA MODEL GROUP NAME}/{DATA MODEL NAME}* ***IP**: IP address of the mediation gateway* ***Port**: port address of the mediation gateway running* ***DATA MODEL GROUP NAME}/DATA MODEL NAME** : check http://{IP}:{PORT}//FiwareDataModel* *example:* • *http://localhost:8081/Subscribe/Test/Room* • *http://localhost:8081/Subscribe/Test/Car*

	• http://localhost:8081/Subscribe/Farm/Building

Sample subscription body	{
	"description": "A subscription to get info about Room8",
	"subject": {
	"entities": [
	{
	"id": "Room8",
	"type": "Room"
	}
	],
	"condition": {
	"attrs": [
	"pressure",
	"temperature"
	]
	}
	},
	"notification": {
	"http": {
	"url": "http://143.248.57.28:8081/Subscribe/Test/Room"
	},
	"attrs": [
	"pressure",
	"temperature"
	]
	},
	"expires": "2040-01-01T14:00:00.00Z",
	"throttling": 5
	}

You can use the mediation gateway to generate sample



5. Check if there is epcis event related to Room 8

Note: Note: During translation sample key is generated as follows urn:epc:id:sgtin:88000269.[entityID]

Server	[EPCIS] localhost:8080
Method	GET
URL	http://localhost:8080/epcis/Service/Poll/SimpleEventQuery?MATCH_epc=urn:epc:id:sgtin:88000269.Room8
Status	200 OK
Response	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <EPCISQueryDocumentType xmlns:ns2="http://www.unece.org/cefact/namespaces /StandardBusinessDocumentHeader" xmlns:ns4="urn:epcglobal:epcis:xsd:1" xmlns:ns3="urn:epcglobal:epcis-query:xsd:1"> <EPCISBody> <ns3:QueryResults> <queryName>SimpleEventQuery</queryName> <resultsBody> <EventList/> </resultsBody> </ns3:QueryResults> </EPCISBody> </EPCISQueryDocumentType>
Comment	It returns empty event list

6. Subscribe

Server	[FIWARE] localhost:1026/v2
Method	POST
URL	localhost:1026/v2/ subscriptions
Headers	Content-Type: application/json

Body	{
	"description": "A subscription to get info about Room8",
	"subject": {
	"entities": [
	{
	"id": "Room8",

	"type": "Room"
	}
	],
	"condition": {
	"attrs": [
	"pressure",
	"temperature"
	]
	}
	},
	"notification": {
	"http": {
	"url": "http://143.248.57.28:8081/Subscribe/Test/Room"
	},
	"attrs": [
	"pressure",
	"temperature"
	]
	},
	"expires": "2040-01-01T14:00:00.00Z",
	"throttling": 5
	}

Status	201 Created
Response	{}
Comment	Subscription to Entity Room8 is created

7. Check if there is epcis event related to Room 8 after the subscription

Server	[EPCIS] localhost:8080
Method	GET
URL	http://localhost:8080/epcis/Service/Poll/SimpleEventQuery?MATCH_epc=urn:epc:id:sgtin:88000269.Room8
Status	200 OK
Response	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <EPCISQueryDocumentType xmlns:ns2="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader" xmlns:ns4="urn:epcglobal:epcis:xsd:1" xmlns:ns3="urn:epcglobal:epcis-query:xsd:1"> <EPCISBody> <ns3:QueryResults> <queryName>SimpleEventQuery</queryName> <resultsBody> <EventList> <ObjectEvent> <eventTime>2018-08-28T17:22:09.363Z</eventTime>

	<recordTime>2018-08-28T17:22:09.417Z</recordTime> <eventTimeZoneOffset>-05:00</eventTimeZoneOffset> <baseExtension> <eventID>4829cb2a-97a9-43fd-bf31-fb0374a7c792</eventID> </baseExtension> <epcList> <epc>urn:epc:id:sgtin:88000269.Room8</epc> </epcList> <action>OBSERVE</action> <bizStep>urn:epcglobal:cbv:bizstep:driving</bizStep> <disposition>urn:epcglobal:cbv:disp:on_the line</disposition> <readPoint> <id>urn:epc:id:sgln:8800026900016.Room8</id> </readPoint> <bizLocation> <id>urn:epc:id:sgln:8800026900016.103.Room8</id> </bizLocation> <bizTransactionList> <bizTransaction type="urn:epcglobal:cbv:Bus:status">http://transaction.acme.com/po/urn:epcglobal:cbv:bizstep:Sensing</bizTransaction> </bizTransactionList> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/id">Room8</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/type">Room</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature"> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature/type">Float</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature/value">28.0</oliot:Fiware> </oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/Pressure"> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/Pressure/type">Float</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/pressure/value">28.0</oliot:Fiware> </oliot:Fiware> </ObjectEvent> </EventList> </resultsBody> </ns3:QueryResults> </EPCISBody> </EPCISQueryDocumentType>
Comment	One event is returned

8. Update any value of the Room

Server	[FIWARE] localhost:1026/v2

Method	PATCH
URL	localhost:1026/v2/entities/Room8/attrs
Headers	Accept: application/json Content-Type: application/json

Body	{
	"pressure": {
	"type": "Integer",
	"value": 123,
	"metadata": {}
	},
	"temperature": {
	"type": "Float",
	"value": 40,
	"metadata": {}
	}
	}

Status	204 No Content
Comment	Temperature value of Entity Room8 is updated to 40

9. Check if there are two Room8 events are created in epcis

Server	[EPCIS] localhost:8080
Method	GET
URL	http://localhost:8080/epcis/Service/Poll/SimpleEventQuery?MATCH_epc=urn:epc:id:sgtin:88000269.Room8
Status	200 OK
Response	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <EPCISQueryDocumentType xmlns:ns2="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader" xmlns:ns4="urn:epcglobal:epcis:xsd:1" xmlns:ns3="urn:epcglobal:epcis-query:xsd:1"> <EPCISBody> <ns3:QueryResults> <queryName>SimpleEventQuery</queryName> <resultsBody> <EventList> <ObjectEvent> <eventTime>2018-08-28T17:22:09.363Z</eventTime> <recordTime>2018-08-28T17:22:09.417Z</recordTime> <eventTimeZoneOffset>-05:00</eventTimeZoneOffset> <baseExtension> <eventID>4829cb2a-97a9-43fd-bf31-fb0374a7c792</eventID> </baseExtension> <epcList> <epc>urn:epc:id:sgtin:88000269.Room8</epc> </epcList> <action>OBSERVE</action> <bizStep>urn:epcglobal:cbv:bizstep:driving</bizStep>

<disposition>urn:epcglobal:cbv:disp:on_the line</disposition>
<readPoint>
<id>urn:epc:id:sgln:8800026900016.Room8</id>
</readPoint>
<bizLocation>
<id>urn:epc:id:sgln:8800026900016.103.Room8</id>
</bizLocation>
<bizTransactionList>
<bizTransaction
type="urn:epcglobal:cbv:Bus:status">http://transaction.acme.com/po/urn:epcglobal:cbv:
bizstep:Sensing</bizTransaction>
</bizTransactionList>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/id">Room8</oliot:Fiware>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/type">Room</oliot:Fiware>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature">
<oliot:Fiware
xmlns:oliot="http://ns.oliot.com/temprature/type">Float</oliot:Fiware>
<oliot:Fiware
xmlns:oliot="http://ns.oliot.com/temprature/value">28.0</oliot:Fiware>
</oliot:Fiware>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/Pressure">
<oliot:Fiware
xmlns:oliot="http://ns.oliot.com/Pressure/type">Float</oliot:Fiware>
<oliot:Fiware
xmlns:oliot="http://ns.oliot.com/pressure/value">28.0</oliot:Fiware>
</oliot:Fiware>
</ObjectEvent>
<ObjectEvent>
<eventTime>2018-08-28T17:32:24.632Z</eventTime>
<recordTime>2018-08-28T17:32:24.689Z</recordTime>
<eventTimeZoneOffset>-05:00</eventTimeZoneOffset>
<baseExtension>
<eventID>13baa91d-44ac-4a9e-a7ac-b10d6a10a464</eventID>
</baseExtension>
<epcList>
<epc>urn:epc:id:sgtin:88000269.Room8</epc>
</epcList>
<action>OBSERVE</action>
<bizStep>urn:epcglobal:cbv:bizstep:driving</bizStep>
<disposition>urn:epcglobal:cbv:disp:on_the line</disposition>
<readPoint>
<id>urn:epc:id:sgln:8800026900016.Room8</id>
</readPoint>
<bizLocation>
<id>urn:epc:id:sgln:8800026900016.103.Room8</id>
</bizLocation>
<bizTransactionList>
<bizTransaction
type="urn:epcglobal:cbv:Bus:status">http://transaction.acme.com/po/urn:epcglobal:cbv:
bizstep:Sensing</bizTransaction>
</bizTransactionList>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/id">Room8</oliot:Fiware>
<oliot:Fiware xmlns:oliot="http://ns.oliot.com/type">Room</oliot:Fiware>

	<oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature"> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature/type">Float</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/temprature/value">40.0</oliot:Fiware> </oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/Pressure"> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/Pressure/type">Float</oliot:Fiware> <oliot:Fiware xmlns:oliot="http://ns.oliot.com/pressure/value">40.0</oliot:Fiware> </oliot:Fiware> </ObjectEvent> </EventList> </resultsBody> </ns3:QueryResults> </EPCISBody> </EPCISQueryDocumentType>
Comment	One event is returned

Appendix

GS1 Key proposal for farming

Objects to be identified	FIRWARE Key	GS1 Key	Comment
Farm	urn:entity:farm:<farmId>	urn:epc:id:sgln:{companyPrefix}:{locationReference}:{extensionComponent}	SGLN is used here (GIAI can be used) Example: Farm→100 *urn:epc:id:sgln:88000269:100:<farmId>*
Building	urn:entity:building:<buildingId>	urn:epc:id:sgln:{companyPrefix}:{locationReference}:{extensionComponent}	SGLN is used here (GIAI can be used) Example: building →101 *urn:epc:id:sgln:88000269:101:<buildingId >*
Pen	urn:entity:pen:<penId>	urn:epc:id:sgln:{companyPrefix}:{locationReference}:{extensionComponent}	SGLN is used here (GIAI can be used) Example: building →102 *urn:epc:id:sgln:88000269:101:<penId >*
Pig	urn:entity:pig:<pigId>	urn:epc:id:sgtin:{companyPrefix}:{ItemReference}:{SerialNumber}	GTIN is used here Example: building →103 *urn:epc:id:sgtin:88000269:101:<pigId >*
slaughterhouse	urn:entity:slaughterhouse:<slaughterhouseId>	urn:epc:id:sgln:{companyPrefix}:{locationReference}:{extensionComponent}	SGLN is used here (GIAI can be used) Example: building →104 *urn:epc:id:sgln:88000269:104:<slaughterhouseId >*

FIWARE Pig Farming Examples

Pig example


```
{
    "id": "Pig-b5b4da51-373d-444c-ab74-fe27109f3d83",
    "type": "Pig",
    "additionalInfo": {
        "type": "T",
        "value": {
            "ILVOPenId": "7",
            "feed_intake": "329",
            "visit_time": "2018-05-08 19:53:14",
            "ILVOPeriod": "2",
            "weight": "31500",
            "responder": "984000100625782",
            "ILVONRstation": "5782",
            "ILVOLFtag": "984000100625782",
            "duration": "717",
            "ILVOPigId": "10",
            "DEBUG-filename": "tempRealIlvoNedapVelosvelos_ilvo.vpu-online.com_ppt_location0-9999_2018-05-
08.csv",
            "ILVOHFtagLeft": "E00401005BA42B5A",
            "lifenumber": "10",
            "location": "7",
            "state": "0",
```

```
            "DEBUG-currentLine": "552",
            "ILVOHFtagRight": "E00401005BA42C4F",
            "ILVOSanitel": "70148"
        },
        "metadata": {}
    },
    "arrivalTimestamp": {
        "type": "Text",
        "value": "",
        "metadata": {}
    },
    "buildingId": {
        "type": "Text",
        "value": "",
        "metadata": {}
    },
    "companyId": {
        "type": "Text",
        "value": "8b6e0aa4-08fc-4f6f-960d-5a65a748b0e7",
        "metadata": {}
    },
    "compartmentId": {
        "type": "Text",
        "value": "",
        "metadata": {}
    },
    "endTimestampAcquisition": {
        "type": "Number",
        "value": 1534895999,
        "metadata": {}
    },
    "endTimestampMonitoring": {
        "type": "Number",
        "value": 1534895999,
        "metadata": {}
    },
    "farmId": {
        "type": "Text",
        "value": "9a68ea4e-348e-424e-9346-6e9fefaf18db",
        "metadata": {}
    },
    "lastUpdate": {
        "type": "DateTime",
        "value": "2018-05-08T19:53:14.00Z",
        "metadata": {}
    },
    "penId": {
```

```
        "type": "Text",
        "value": "b1d7f4b0-2d8f-4c6d-b7f4-ed03e4ed25ee",
        "metadata": {}
    },
    "pigId": {
        "type": "Text",
        "value": "b5b4da51-373d-444c-ab74-fe27109f3d83",
        "metadata": {}
    },
    "serialNumber": {
        "type": "Text",
        "value": "",
        "metadata": {}
    },
    "sex": {
        "type": "Text",
        "value": "B",
        "metadata": {}
    },
    "startTimestampAcquisition": {
        "type": "Number",
        "value": 1519430400,
        "metadata": {}
    },
    "startTimestampMonitoring": {
        "type": "Number",
        "value": 1519430400,
        "metadata": {}
    },
    "totalConsumedFood": {
        "type": "Number",
        "value": 329,
        "metadata": {}
    },
    "totalConsumedWater": {
        "type": "Text",
        "value": "",
        "metadata": {}
    },
    "totalTimeConsumedFood": {
        "type": "Number",
        "value": 717,
        "metadata": {}
    },
    "totalTimeConsumedWater": {
        "type": "Text",
        "value": "",
```

```
            "metadata": {}
        },
        "weight": {
            "type": "Number",
            "value": 31500,
            "metadata": {}
        }
    }
```


Building Example


```
{
        "id": "Building-319cba0c-773d-4964-b750-136a8d5fb3c1",
        "type": "Building",
        "CO2": {
            "type": "Text",
            "value": "",
            "metadata": {}
        },
        "additionalInfo": {
            "type": "T",
            "value": {
                "SensorState": "null",
                "GatewayID": "904987",
                "CheckDigit": "KOIP",
                "accountID": "16892",
                "PlotValues": "5",
                "monnitApplicationID": "21",
                "powerSourceID": "2",
                "Voltage": "2.89",
                "Battery": "72",
                "lastCommunicationDate": "10/16/2019 2:00:02 PM",
                "applicationID": "21",
                "ApplicationID": "null",
                "batteryLevel": "72",
                "SignalStrength": "94",
                "SensorID": "324246",
                "PlotValue": "5",
                "canUpdate": "True",
                "MessageDate": "10/16/2019 8:00:02 PM",
                "sensorName": "Lux - 324246 - Hok 4 AP",
                "signalStrength": "96",
```

```
            "Data": "5",
            "plotLabels": "Lux",
            "sensorID": "324246",
            "currentReading": "10 lux",
            "AlertSent": "null",
            "nextCommunicationDate": "10/16/2019 2:10:02 PM",
            "MetNotificationRequirements": "False",
            "DataValues": "5",
            "CSNetID": "27898",
            "State": "0",
            "alertsActive": "True",
            "DisplayData": "5 lux",
            "lastDataMessageMessageGUID": "7a3b9ec6-db5b-4d84-a254-e7a77bf3f036",
            "status": "0",
            "DataMessageGUID": "75939fd1-51bc-47ce-b69d-9e2faf8fffae",
            "DataTypes": "LuxData"
        },
        "metadata": {}
    },
    "buildingId": {
        "type": "Text",
        "value": "319cba0c-773d-4964-b750-136a8d5fb3c1",
        "metadata": {}
    },
    "companyId": {
        "type": "Text",
        "value": "8b6e0aa4-08fc-4f6f-960d-5a65a748b0e7",
        "metadata": {}
    },
    "farmId": {
        "type": "Text",
        "value": "9a68ea4e-348e-424e-9346-6e9fefaf18db",
        "metadata": {}
    },
    "humidity": {
        "type": "Number",
        "value": 81.39,
        "metadata": {}
    },
    "lastUpdate": {
        "type": "DateTime",
        "value": "2019-10-16T20:00:02.00Z",
        "metadata": {}
    },
    "luminosity": {
        "type": "Number",
        "value": 5,
```

```
        "metadata": {}
    },
    "name": {
        "type": "Text",
        "value": "ILVO building 1",
        "metadata": {}
    },
    "temperature": {
        "type": "Number",
        "value": 22.97,
        "metadata": {}
    }
}
```


FIWARE data models schema for farm

Farm Entity


```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Farm",
  "type": "object",
  "properties": {
          "farmId":{
                  "type": "Text",
                  "description": "It represents the id of the Farm Entity (the <farmId> contained in the EntityId)"
  },
          "type":{
                  "type": "Text",
                  "value": "Farm",
                  "description": "Entity Type"
  },
  "address": {
    "type": "Text",
    "description": "It represents the address of the farm",
          "metadata":{}
  },
  "name": {
    "type": "Text",
    "description": "It represents the name of the farm",
          "metadata":{}
  },
  "ownerCompany": {
    "type": "Text",
    "description": "It represents the name of the company that owns the farm",
          "metadata":{}
  }
```

```
  }
}

{
        "farmId":"urn:entity:farm:<farmID>",
        "type":"Farm",
        "address":"La Cañada 04120 Almería Spain",
        "name":"Greenhouse agriculture",
        "ownerCompany":"Maria"

}
```

Building Entity

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Building",
  "type": "object",
  "properties": {
        "buildingId":{
                "type": "Text",
                "description": "It represents the id of the Building Entity (the <buildingId> contained in the
EntityId attribute)"
    },
        "type":{
                "type": "Text",
                "value": "Building",
                "description": "Entity Type"
    },
    "name": {
     "type": "Text",
     "description": "It represents the name of the building",
          "metadata":{}
    },
    "lastUpdate": {
     "type": "DateTime",
     "description": "It represents the timestamp of the last update",
          "metadata":{}
    },
    "farmId": {
     "type": "Text",
     "description": "It represents the id of the Farm in which the Building is located (the farmId)",
          "metadata":{}
    },
    "temperature": {
     "type": "Float",
     "description": "It represents the last value of the temperature registered within the Building",
          "metadata":{
           "uom": {
```

```
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#DegreeCelsius"
          }
           }
  },
  "humidity": {
   "type": "Float",
   "description": "It represents the last value of the humidity registered within the Building",
           "metadata":{
            "uom": {
             "type": "string",
             "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Humidity"
          }
           }
  },
  "luminosity": {
   "type": "Float",
   "description": "It represents the last value of the luminosity registered within the Building",
           "metadata":{
            "uom": {
             "type": "string",
             "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#LuminousIntensity"
          }
           }
  }
 }
}

{
          "buildingId":"urn:entity:building:<buildingId>",
          "type":"Building",
          "name":{
       "type": "Text",
       "value": "La Cañada 04120 Almería Spain",
       "metadata": {}
  },
          "lastUpdate":{
       "type": "ISO8601",
       "value": "2018-08-22T05:10:58.00Z",
       "metadata": {}
  },
          "farmId":"urn:entity:farm:<farmID>",
          "temperature":{
       "type": "Float",
       "value": 37.6,
       "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#DegreeCelsius"
                                        }
                                }
  },
          "humidity":{
```

```
        "type": "Float",
        "value": 45,
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Humidity"
                                        }
                                }
   },
          "luminosity":{
        "type": "Float",
        "value": 0.6,
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#LuminousIntensity"
                                        }
                                }
   }
}
```

Pen Entity

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Pig",
  "type": "object",
  "properties": {
          "pigId":{
                  "type": "Text",
                  "description": "It represents the id of the Pig Entity (the <pigId> contained in the EntityId
attribute)"
   },
          "type":{
                  "type": "Text",
                  "value": "Pig",
                  "description": "Entity Type"
   },
   "serialNumber": {
    "type": "Text",
    "description": "If a serial number is assigned to the pig by the farm, this field contains such a value",
          "metadata":{}
   },
   "lastUpdate": {
    "type": "DateTime",
    "description": "It represents the timestamp of the last update",
          "metadata":{}
   },
   "penId": {
```

```
    "type": "Text",
    "description": "It represents the id of the Farm in which the pen is located (the penId)",
            "metadata":{}
   },
   "weight": {
    "type": "Float",
    "description": "It represents the current weight of the pig (the last measured value)",
            "metadata":{
             "uom": {
              "type": "string",
              "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
           }
            }
   },
   "totalConsumedWater": {
    "type": "Float",
    "description": "it represents the amount of water that was consumed between the moment in
                                                which the pig started to drink and the current moment (e.g., if the
pig started
                                                to drink 2 minutes ago and is continuing to drink, this value
contains the
                                                total amount of water that the pig drunk since 2 minutes ago)",
            "metadata":{
             "uom": {
              "type": "string",
              "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Litre"
           }
            }
   },
   "totalConsumedFood": {
    "type": "Float",
    "description": "it represents the amount of food that was consumed between the moment
                                                        in which the pig started to eat and the current moment
(e.g., if the
                                                        pig started to eat 2 minutes ago and is continuing to eat,
this value
                                                        contains the total amount of food that the pig ate since 2
minutes ago)",
            "metadata":{
             "uom": {
              "type": "string",
              "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
           }
            }
   }
  }
}

{
           "pigId":"urn:entity:pig:<pigId>",
           "type":"Pig",
           "serialNumber":{
         "type": "Text",
```

```
        "value": "8764321000003",
        "metadata": {}
    },
            "lastUpdate":{
        "type": "ISO8601",
        "value": "2018-08-22T05:10:58.00Z",
        "metadata": {}
    },
            "penId":"urn:entity:pen:<penId>",
            "weight":{
        "type": "Float",
        "value": 37.6,
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
                                        }
                                }
    },
            "totalConsumedWater":{
        "type": "Float",
        "value": 20,
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Litre"
                                        }
                                }
    },
            "totalConsumedFood":{
        "type": "Float",
        "value": 45,
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
                                        }
                                }
    }
}
```

Pig Entity

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Pig",
  "type": "object",
  "properties": {
            "pigId":{
```

```
                    "type": "Text",
                    "description": "It represents the id of the Pig Entity (the <pigId> contained in the EntityId
attribute)"
   },
          "type":{
                    "type": "Text",
                    "value": "Pig",
                    "description": "Entity Type"
   },
   "serialNumber": {
    "type": "Text",
    "description": "If a serial number is assigned to the pig by the farm, this field contains such a value",
           "metadata":{}
   },
   "lastUpdate": {
    "type": "DateTime",
    "description": "It represents the timestamp of the last update",
           "metadata":{}
   },
   "penId": {
    "type": "Text",
    "description": "It represents the id of the Farm in which the pen is located (the penId)",
           "metadata":{}
   },
   "weight": {
    "type": "Float",
    "description": "It represents the current weight of the pig (the last measured value)",
           "metadata":{
            "uom": {
             "type": "string",
             "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
          }
           }
   },
   "totalConsumedWater": {
    "type": "Float",
    "description": "it represents the amount of water that was consumed between the moment in
                                                which the pig started to drink and the current moment (e.g., if the
pig started
                                                to drink 2 minutes ago and is continuing to drink, this value
contains the
                                                total amount of water that the pig drunk since 2 minutes ago)",
           "metadata":{
            "uom": {
             "type": "string",
             "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Litre"
          }
           }
   },
   "totalConsumedFood": {
    "type": "Float",
    "description": "it represents the amount of food that was consumed between the moment
```

```
                                                    in which the pig started to eat and the current moment
(e.g., if the
                                                    pig started to eat 2 minutes ago and is continuing to eat,
this value
                                                    contains the total amount of food that the pig ate since 2
minutes ago)",
              "metadata":{
               "uom": {
                "type": "string",
                "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
              }
               }
   }
  }
}

{
           "pigId":"urn:entity:pig:<pigId>",
           "type":"Pig",
           "serialNumber":{
         "type": "Text",
         "value": "8764321000003",
         "metadata": {}
   },
           "lastUpdate":{
         "type": "ISO8601",
         "value": "2018-08-22T05:10:58.00Z",
         "metadata": {}
   },
           "penId":"urn:entity:pen:<penId>",
           "weight":{
         "type": "Float",
         "value": 37.6,
         "metadata": {
                                        "uom": {
              "type": "string",
              "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
                                        }
                              }
   },
           "totalConsumedWater":{
         "type": "Float",
         "value": 20,
         "metadata": {
                                        "uom": {
              "type": "string",
              "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Litre"
                                        }
                              }
   },
           "totalConsumedFood":{
         "type": "Float",
         "value": 45,
```

```
        "metadata": {
                                        "uom": {
            "type": "string",
            "value": "http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl#Kilogram"
                                        }
                                }
   }
}
```

Slaughteredpig Entity

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "SlaughteredPig",
  "type": "object",
  "properties": {
          "lastSlaughteredPigId":{
                  "type": "Text",
                  "description": "It represents the id of the last Pig that was slaughtered"
   },
          "type":{
                  "type": "Text",
                  "value": "SlaughteredPig",
                  "description": "Entity Type"
   },
   "serialNumber": {
    "type": "Text",
    "description": "It represents the serialNumber (if it exists) of the last Pig that was slaughtered",
           "metadata":{}
   },
   "lastUpdate": {
    "type": "DateTime",
    "description": "It represents the timestamp of the last update",
           "metadata":{}
   },
   "slaughterhouseId": {
    "type": "Text",
    "description": "It represents the id of the slaughterehouse in which the Pig was slaughtered",
           "metadata":{}
   }
  }
}

{
          "lastSlaughteredPigId":"urn:entity:pig:<pigId>",
          "type":"Pig",
          "serialNumber":{
        "type": "Text",
        "value": "8764321000003",
```

```
        "metadata": {}
   },
            "lastUpdate":{
        "type": "ISO8601",
        "value": "2018-08-22T05:10:58.00Z",
        "metadata": {}
   },
            "slaughterhouseId":"urn:entity:slaughterhouse:<slaughterhouseId>",
}
```

Slaughterhouse Entity

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "slaughterhouse",
  "type": "object",
  "properties": {
            "slaughterhouseId":{
                        "type": "Text",
                        "description": "it represents the id of the slaughterhouse Entity (the <slaughterhouseId>
contained in the EntityId attribute)"
   },
            "type":{
                        "type": "Text",
                        "value": "Slaughterhouse",
                        "description": "Entity Type"
   },
            "address:": {
     "type": "Text",
     "description": "It represents the address of the slaughterhouse",
              "metadata":{}
   },
            "name": {
     "type": "Text",
     "description": "It represents the name of the building",
              "metadata":{}
   }
  }
}



{
            "slaughterhouseId":"urn:entity:slaughterhouse:<slaughterhouseId>",
            "type":"Slaughterhouse",
            "address":{
        "type": "Text",
        "value": "La Cañada 04120 Almería Spain",
        "metadata": {}
```

```
   },
            "name":{
         "type": "Text",
         "value": "pig slaughterhouse",
         "metadata": {}
   }
}
```

Entity List

```
{
  "$id": "https://resl.com/farm.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "EntityList",
  "type": "object",
  "properties": {
            "list":{
                        "type": "array",
                        "maxItems": 3,
                        "items": {
        "type": "string"
      }
            },
            "type":{
                        "type": "Text",
                        "value": "EntityList",
                        "description": "Entity Type"
   },
            "lastUpdate": {
     "type": "DateTime",
     "description": "It represents the timestamp of the last update",
              "metadata":{}
   },
            "lastAddedItem": {
     "type": "Text",
     "description": "It contains the <id> of the last added <Entity> entity",
              "metadata":{}
   },
            "lastRemovedItem": {
     "type": "Text",
     "description": "It contains the <id> of the last removed <Entity> entity",
              "metadata":{}
   }
  }
}



{
```

```
            "list":["urn:entity:pen:<penId1>", "urn:entity:pen:<penId2>", "urn:entity:pen:<penId3>"],
            "type":"EntityList",
            "lastUpdate":{
        "type": "ISO8601",
        "value": "2018-08-22T05:10:58.00Z",
        "metadata": {}
    },
            "lastAddedItem":{
        "type": "Text",
        "value": "urn:entity:pen:<penId3>",
        "metadata": {}
    },
            "lastRemovedItem":{
        "type": "Text",
        "value": "urn:entity:pen:<penId0>",
        "metadata": {}
    }
}
```


EPCIS data model schema for farm

On progress ….