Open APIs for Open Minds

Creating context historic using Cygnus

Francisco Romero Bueno
Technological Specialist. FIWARE data engineer
francisco.romerobueno@telefonica.com

FIWARE

Before starting…

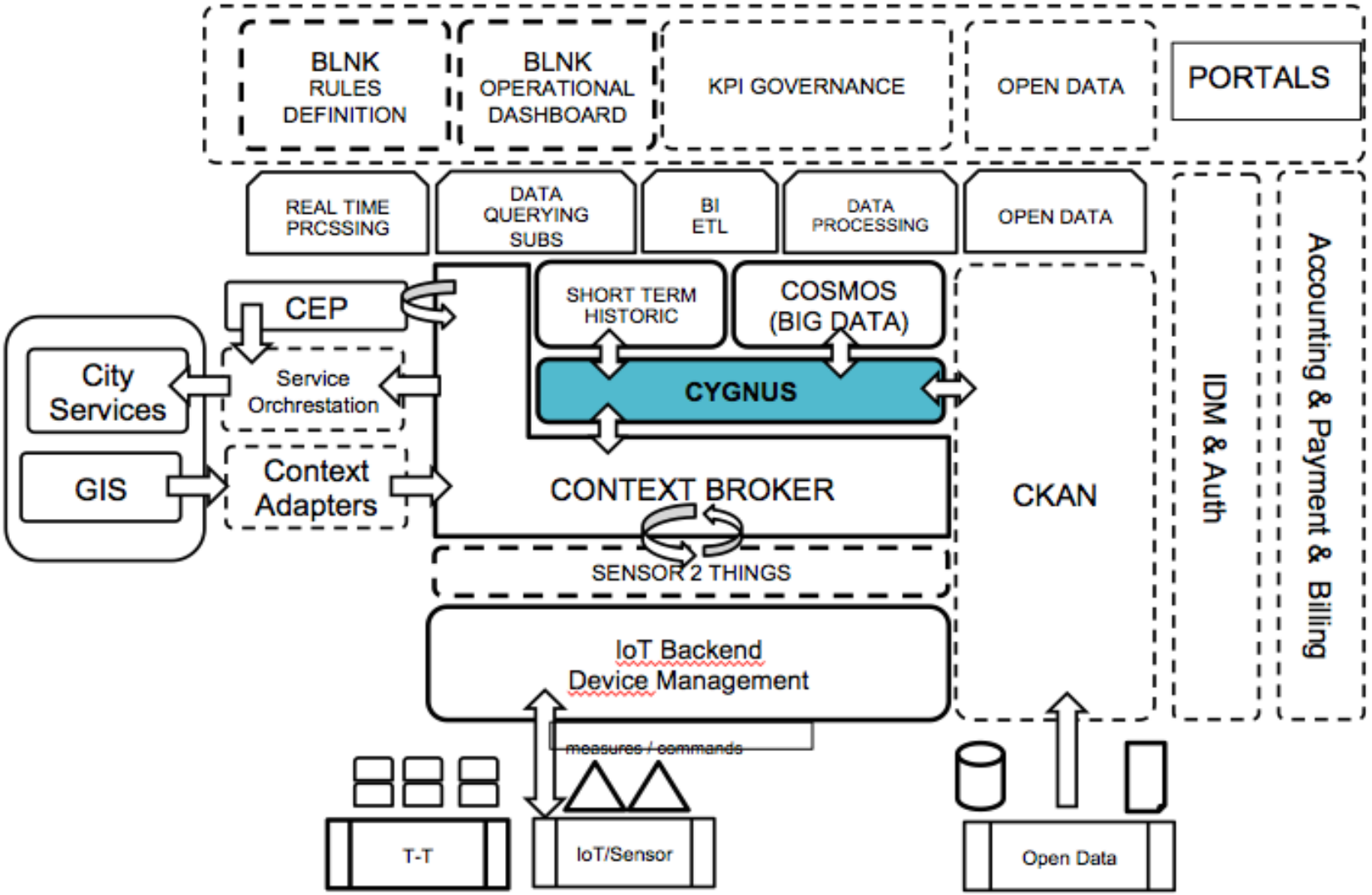
https://github.com/telefonicaid/fiware-cygnus

http://fiware-cygnus.readthedocs.io/en/1.6.0/

https://hub.docker.com/r/fiware/cygnus-ngsi/

http://repositories.testbed.fiware.org/repo/

Cygnus place in FIWARE architecture



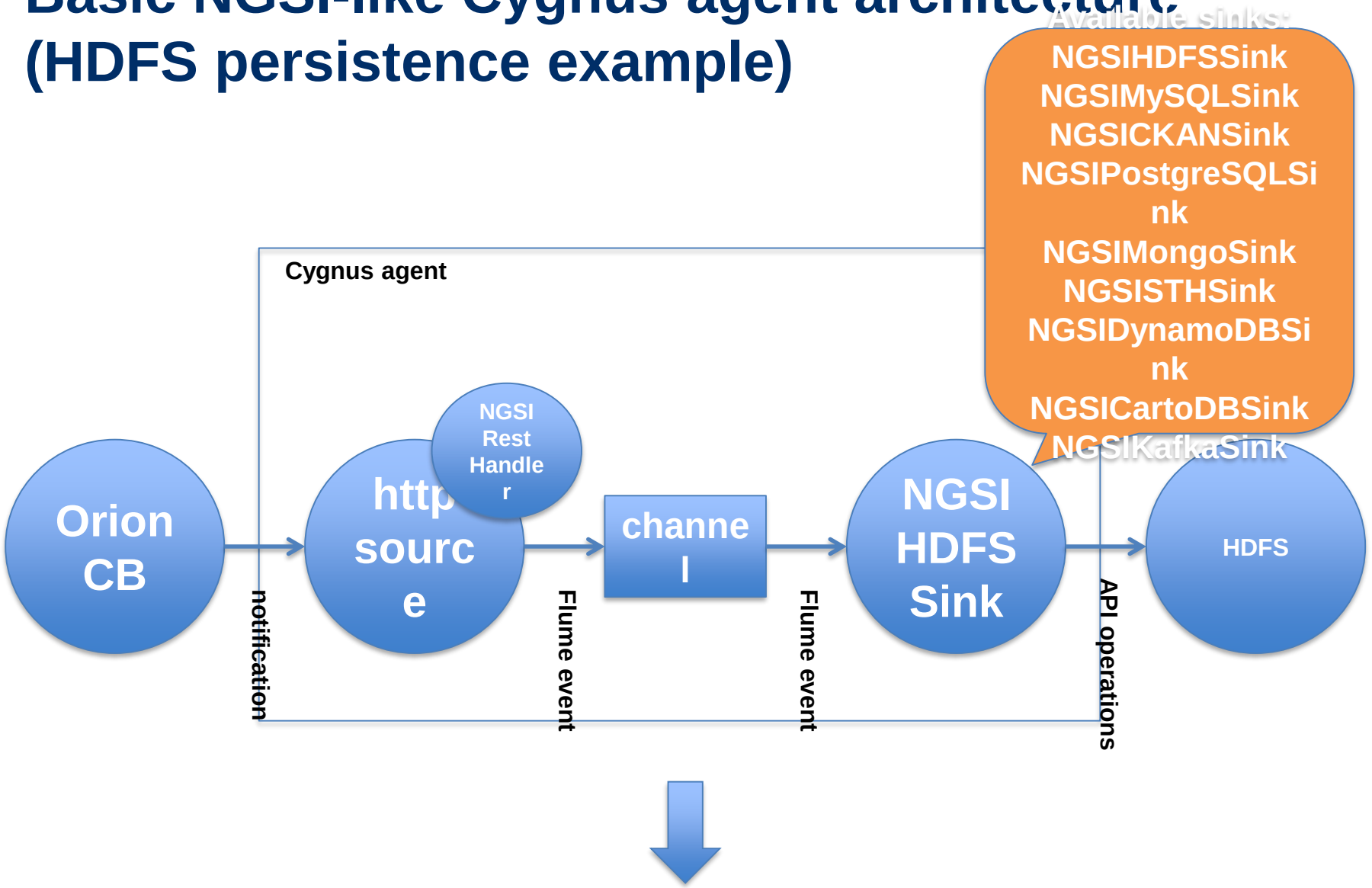
Cygnus FAQ

- **What is it for?**
  - Cygnus is a connector in charge of persisting multiple sources of data (mainly NGSI) in certain configured **third-party storages**, creating a **historical view** of such data.
- **Why creating historics from NGSI context data?**
  - **Orion Context Broker** only stores the last value regarding an entity's attribute, and if an older value is required then you will have to persist it in other storage, value by value, using Cygnus.
- **How does it receives context data from Orion Context Broker?**
  - Cygnus uses the **subscription/notification** feature of Orion. A subscription is made in Orion on behalf of Cygnus, detailing which entities we want to be notified when an update occurs on any of those entities attributes.

Cygnus FAQ

- **Which storages is it able to integrate?**
  - Current release is able to persist Orion context data in:
    - HDFS, the Hadoop distributed file system.
    - MySQL and PostgreSQL, the well-know relational database managers.
    - CKAN, an Open Data platform.
    - MongoDB and STH (Short-Term Historic)
    - Kafka queues
    - DynamoDB, the cloud-based NoSQL database
    - CartoDB, the geolocated platform
- **Which is its architecture?**
  - Internally, Cygnus is based on **Apache Flume**. Thus, **running Cygnus is running a Flume agent**, which is basically composed of a *source* in charge of receiving the data, a *channel* where the source puts the data once it has been transformed into a Flume event, and a *sink*, which takes Flume events from the channel in order to persist the data within its body into a third-party storage.

Basic NGSI-like Cygnus agent architecture (HDFS persistence example)



Cygnus Agent architecture becomes Cygnus Agent configuration

Cygnus agent configuration (simplified HDFS)

```
cygnus.sources=http-source
cygnus.sinks=hdfs-sink
cygnus.channels=hdfs-channel
```

declarations

```
cygnus.sources.http-source.type=http
cygnus.sources.http-source.channels=hdfs-channel
cygnus.sources.http-source.handler=OrionRestHandler
```

common to all sources

```
cygnus.sources.http-source.handler.default_service=default
cygnus.sources.http-source.handler.default_service_path=/
cygnus.sources.http-source.handler.interceptors=timestamp
```

handler specific

```
cygnus.sinks.hdfs-sink.type=OrionHDFSSink
cygnus.sinks.hdfs-sink.channel=hdfs-channel
cygnus.sinks.hdfs-sink.data_model=dm-by-entity
```

common to all sinks

```
cygnus.sinks.hdfs-sink.batch_size=200
cygnus.sinks.hdfs-sink.batch_time=10
cygnus.sinks.hdfs-sink.batch_ttl=5
cygnus.sinks.hdfs-sink.hdfs_host=cosmos.lab.fiware.org
cygnus.sinks.hdfs-sink.hdfs_port=14000
cygnus.sinks.hdfs-sink.hdfs_username=acs
cygnus.sikns.hdfs-sink.oauth2_token=fgskjasfq4fasdsa33
cygnus.sinks.hdfs-sink.file_format=json-column
```

sink specific

```
cygnus.channels.hdfs-channel.type=memory
cygnus.channels.hdfs-channel.capacity=1000
cygnus.channels.hdfs-channel.transactionCapacity=100
```

common to all channels

Some interesting common parameters

- Http sources
  - **type**: always http (alias of org.apache.flume.sources.http)
  - **channels**: channels the events (notifications) are written to
  - **handler**: specializes the generic Http source, making it a NGSI-like Http source
- Memory channels
  - **type**: always memory (alias of org.apache.flume.channels.memory)
  - **capacity**: number of events the channel is able to contain
  - **transactionCapacity**: number of events that can be read per transaction
- Sinks
  - **type**: com.telefonica.iot.cygnus.sinks.NGSIXXXXSink
  - **channel**: channel the events (notifications) are read from
  - **data_model**: how the data is going to be structured in the final storage
  - **attr_persistence**: how the data is going to formated (file_format for HDFS)

Before continuing

- A NGSIv1 notification looks like:

```
POST http://localhost:5050/notify

Content-Type: application/json; charset=utf-8
Accept: application/json
User-Agent: orion/1.2.0
Fiware-Service: vehicles
Fiware-ServicePath: /4wheels

{
   "subscriptionId": "51c0ac9ed714fb3b37d7d5a8",
   "originator": "localhost",
   "contextResponses": [{
      "contextElement": {
         "attributes": [
            {"name": ”speed", "type": ”float", "value": ”112.9", "metadatas": []},
            {“name”: “oil_level", "type": ”float", "value": ”74.6", "metadatas": []}
         ],
         "type": ”car",
         "isPattern": "false",
         "id": ”car1”
      },
      "statusCode": {
         "code": "200",
         "reasonPhrase": "OK”
      }
   }]
}
```

NGSIHDFSSink data model

- Data is stored within HDFS files, under certain HDFS folders
  - The only accepted data_model is dm-by-entity

	dm-by-entity
HDFS folder	hdfs://<namenode>/user/<hdfs_user>/<fiware-service>/<fiware-servicePath>/<destination>/
HDFS file	<destination>.txt

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Custom destinations (an even service paths) can be specified by using the Grouping Rules mechanism
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/grouping_rules_conf/index.html

FIWARE

NGSIHDFSSink file formats

- Supported file formats
  - json-row
  - json-column

  } **JavaScript Object Notation**

  - csv-row
  - csv-column

  } **Comma Separated Values**
- row means a line per entity's attribute
- column means a line for all entity's attributes
- row vs. column
  - row-like modes don't rquire the apps to know the number of attributes nor their names an entity has
  - row-like modes don't show nulls on the persisted files
  - row-like modes require more disk space
  - Both formats can be automatically provisioned by Cygnus

NGSIHDFSSink example

hdfs:///user/myuser/vehicles/4wheels/car1_car/car1_car.txt (json-row)

```
{"recvTimeTs":"1429535775","recvTime":"2015-04-
20T12:13:22.41.124Z","fiwareServicePath":"/4wheels","entityId":"car1","entityType":"car","
attrName":"speed","attrType":"float","attrValue":"112.9","attrMd":[]}
{"recvTimeTs":"1429535775","recvTime":"2015-04-
20T12:13:22.41.124Z","fiwareServicePath":"4wheels","entityId":"car1","entityType":"car","a
ttrName":"oil_level","attrType":"float","attrValue":"74.6","attrMd":[]}
...
```

hdfs:///user/myuser/vehicles/4wheels/car1_car/car1_car.txt (json-column)

```
{"recvTime":"2015-04-
20T12:13:22.41.124Z","fiwareServicePath":"/4wheels","entityId":"car1","entityType":"car","
speed":"112.9","speed_md":[],"oil_level":"74.6","oil_level_md":[]}
...
```

hdfs:///user/myuser/vehicles/4wheels/car1_car/car1_car.txt (csv-row)

```
1429535775,2015-04-
20T12:13:22.41.124Z,/4wheels,car1,car,speed,float,112.9,hdfs:///user/myuser/vehicles/4whee
ls/car1_car_speed_float/car1_car_speed_float.txt 1429535775,2015-04-
20T12:13:22.41.124Z,4wheels,car1,car,oil_level,float,74.6,hdfs:///user/myuser/vehicles/4wh
eels/car1_car_oil_level_float/car1_car_oil_level_float.txt
```

hdfs:///user/myuser/vehicles/4wheels/car1_car/car1_car.txt (csv-column)

```
2015-04-
20T12:13:22.41.124Z,112.9,/4wheels,car1,car,hdfs:///user/myuser/vehicles/4wheels/car1_car_
speed_float/car1_car_speed_float.txt,74.6,hdfs:///user/myuser/vehicles/4wheels/car1_car_oi
l_level_float/car1_car_oil_level_float.txt}
...
```

NGSIMySQLSink data model

- Data is stored within MySQL tables, under certain MySQL databases
  - Database and table names depend on the configured data_model

	dm-by-service-path	dm-by-entity
Database name	<fiware-service>	<fiware-service>
Table name	<fiware-servicePath>	<fiware-servicePath>_ <destination>

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Custom destinations (an even service paths) can be specified by using the Grouping Rules mechanism
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/grouping_rules_conf/index.html

NGSIMySQLSink table formats

- Supported table formats
  - row
  - column
- row means a record per entity's attribute
- column means a record for all entity's attributes
- row vs. column
  - row-like mode don't rquire the apps to know the number of attributes nor their names an entity has
    - Cygnus is able to provision the tables in row mode
  - row-like mode don't show nulls on the persisted files
  - row-like mode require more disk space

NGSIMySQLSink example

row mode, dm-by-entity

```
mysql> select * from vehicles.4wheels_car1_car;
+------------+-----------------------------+------------------+----------+----------
--+-------------+----------+----------+--------+ | recvTimeTs | recvTime |
fiwareServicePath | entityId | entityType | attrName | attrType | attrValue | attrMd
| +------------+-----------------------------+------------------+----------+--------
----+-------------+----------+----------+--------+ | 1429535775 | 2015-04-
20T12:13:22.41.124 | 4wheels | car1 | car | speed | float | 112.9 | [] | |
1429535775 | 2015-04-20T12:13:22.41.124 | 4wheels | car1 | car | oil_level | float |
74.6 | [] | +------------+-----------------------------+------------------+---------
-+------------+-------------+----------+----------+--------+
2 row in set (0.00 sec)
```

column mode, dm-by-entity

```
mysql> select * from vehicles.4wheels_car1_car;
+-----------------------------+------------------+----------+------------+-------+--
--------+----------+--------------+ | recvTime | fiwareServicePath | entityId |
entityType | speed | speed_md | oil_level | oil_level_md | +-----------------------
----+------------------+----------+------------+-------+----------+-----------+----
----------+ | 2015-04-20T12:13:22.41.124 | 4wheels | car1 | car | 112.9 | [] | 74.6
| [] | +-----------------------------+------------------+----------+------------+---
----+----------+-----------+--------------+
1 row in set (0.00 sec)
```


FIWARE

NGSICKANSink data model

- Data is stored within CKAN resources, under certain CKAN packages, under certain CKAN organizations
  - The only accepted data_model is dm-by-entity

	dm-by-entity
Organization	<fiware-service>
Package	<fiware_service>_<fiware-servicePath>
Resource	<destination>

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Custom destinations (an even service paths) can be specified by using the Grouping Rules mechanism
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/grouping_rules_conf/index.html

FIWARE

NGSICKANSink resource formats

- Supported table formats
  - row
  - column
- row means a record per entity's attribute
- column means a record for all entity's attributes
- row vs. column
  - row-like mode don't rquire the apps to know the number of attributes nor their names an entity has
    - Thus, Cygnus is able to provision the resources in row mode
  - row-like mode don't show nulls on the persisted files
  - row-like mode require more disk space

NGSICKANSink example

row mode

```
$ curl -s -S -H "Authorization: myapikey"
"http://host:port/api/3/action/datastore_search?r
esource_id=12345"

{
  "help": "Search a DataStore resource...",
  "success": true,
  "result": {
    "resource_id": "12345",
    "fields": [...],
    "records": [
      {
        "entityId": "car1",
        "entityType": "car",
        "fiwareServicePath": "4wheels",
        "attrType": "float",
        "recvTime": "2015-04-20T12:13:22...",
        "recvTimeTs": 1429535775,
        "attrMd": null,
        "attrValue": "112.9",
        "attrName": "speed",
        "_id": 1
      },
      {
        "entityId": "car1",
        "entityType": "car",
        "fiwareServicePath": "4wheels",
        "attrType": "float",
        "recvTime": "2015-04-20T12:13:22...",
        "recvTimeTs": 1429535775,
        "attrMd": null,
        "attrValue": "74.6",
        "attrName": "oil_level",
        "_id": 2
      }
```

column mode

```
$ curl -s -S -H "Authorization: myapikey"
"http://host:port/api/3/action/datastore_search?r
esource_id=12345"

{
  "help": "Search a DataStore resource...",
  "success": true,
  "result": {
    "resource_id": "12345",
    "fields": [...],
    "records": [
      {
        "recvTime": "2015-04-20T12:13:22...",
        "fiwareServicePath": "4wheels",
        "entityId": "car1",
        "entityType": "car",
        "speed": "112.9",
        "speed_md": null,
        "oil_level": "74.6",
        "oil_level_md": null,
        "_id": 1
      }
    ],
    "_links": {...},
    "total": 1
  }
}
```

NGSIMongoSink data model

- Data is stored within MongoDB collections, under certain MongoDB databases
  - Database and collection names depend on the configured data_model (dm-by-service-path, dm-by-entity or dm-by-attribute)

	dm-by-service-path	dm-by-entity	dm-by-attribute
Database name	<prefix><service>	<prefix><service>	<prefix><service>
Collection name	<prefix><servicePath>	<prefix><servicePath> <destination>	<prefix><servicePath> <destination><attrName>

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Grouping rules are not allowed in NGSIMongoSink

NGSIMongoSink collection formats

- Supported collection formats
  - row
  - column
- row means a record per entity's attribute
- column means a record for all entity's attributes
- row vs. column
  - row-like mode don't rquire the apps to know the number of attributes nor their names an entity has
  - row-like mode don't show nulls on the persisted files
  - row-like mode require more disk space
  - Both formats can be automatically provisioned by Cygnus

NGSIMongoSink example

row mode, dm-by-entity

```
> use vehicles
switched to db vehicles
> db['sth_/4wheels'].find()
{ "_id" : ObjectId("5534d143fa701f0be751db82"), "recvTimeTs": "1402409899391",
"recvTime" : "2015-04-20T12:13:22.41.412Z", "attrName" : "speed", "attrType" :
"float", "attrValue" : "112.9" } { "_id" :
ObjectId("5534d143fa701f0be751db83"), "recvTimeTs": "1402409899391",
"recvTime" : "2015-04-20T12:13:22.41.412Z", "attrName" : "oil_level",
"attrType" : "float", "attrValue" : "74.6" }
```

column mode, dm-by-entity

```
> use vehicles
switched to db vehicles
> db['sth_/4wheels_car1_car'].find()
{"_id" : ObjectId("56337ea4c9e77c1614bfdbb7"), "recvTimeTs": "1402409899391",
"recvTime" : "2015-04-20T12:13:22.41.412Z", "speed" : "112.9", "oil_level" :
"74.6"}
```

NGSISTHSink data model

- Data is stored within MongoDB collections, under certain MongoDB databases
  - Database and collection names depend on the configured data_model (dm-by-service-path, dm-by-entity or dm-by-attribute)

	dm-by-service-path	dm-by-entity	dm-by-attribute
Database name	<prefix><service>	<prefix><service>	<prefix><service>
Collection name	<prefix><servicePath>	<prefix><servicePath> <destination>	<prefix><servicePath> <destination><attrName>

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Grouping rules are not allowed in NGSISTHSink

NGSISTHSink collection formats

- Just one format, STH one
  - Based on prepopulating the collections with initial aggregations that are updated upon notification
- Available aggregations
  - Numeric data
    - Max
    - Min
    - Mean
    - Average
  - String data
    - Occurences
- Aggregations are done for different ranges and origins
  - For each second in a minute
  - For each minute in a hour
  - For each hour in a day
  - For each day in a month
  - For each month in a day

NGSISTHSink example

dm-by-entity

```
> use vehicles
switched to db vehicles
> db['sth_/4wheels'].find()
{ "_id" : { "attrName" : "speed", "origin" : ISODate("2015-04-20T00:00:00Z"), "resolution" : "hour", "range" : "day", "attrType" :
"float" }, "points" : [ { "offset" : 0, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" : 12,
"samples" : 1, "sum" : 112.9, "sum2" : 12746.41, "min" : 112.9, "max" : 112.9 }, ..., { "offset" : 23, "samples" : 0, "sum" : 0, "sum2" :
0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" : "speed", "origin" : ISODate("2015-01-01T00:00:00Z"), "resolution" :
"month", "range" : "year", "attrType" : "float" }, "points" : [ { "offset" : 0, "samples" : 1, "sum" : 0, "sum2" : 0, "min" : 0, "max" : 0 },
..., { "offset" : 3, "samples" : 0, "sum" : 112.9, "sum2" : 12746.41, "min" : 112.9, "max" : 112.9 }, ..., { "offset" : 11, "samples" : 0,
"sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" : "speed", "origin" : ISODate("2015-04-
20T12:13:00Z"), "resolution" : "second", "range" : "minute", "attrType" : "float" }, "points" : [ { "offset" : 0, "samples" : 0, "sum" : 0,
"sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" : 22, "samples" : 1, "sum" : 112.9, "sum2" : 12746.41, "min" : 112.9,
"max" : 112.9 }, ..., { "offset" : 59, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" :
"speed", "origin" : ISODate("2015-04-20T12:00:00Z"), "resolution" : "minute", "range" : "hour", "attrType" : "float" }, "points" : [ {
"offset" : 0, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" : 13, "samples" : 1, "sum" : 112.9,
"sum2" : 12746.41, "min" : 112.9, "max" : 112.9 }, ..., { "offset" : 59, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -
Infinity } ] } { "_id" : { "attrName" : "speed", "origin" : ISODate("2015-04-01T00:00:00Z"), "resolution" : "day", "range" : "month",
"attrType" : "float" }, "points" : [ { "offset" : 1, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" :
20, "samples" : 1, "sum" : 112.9, "sum2" : 12746.41, "min" : 112.9, "max" : 112.9 }, ..., { "offset" : 31, "samples" : 0, "sum" : 0,
"sum2" : 0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" : "oil_level", "origin" : ISODate("2015-04-20T00:00:00Z"),
"resolution" : "hour", "range" : "day", "attrType" : "float" }, "points" : [ { "offset" : 0, "samples" : 0, "sum" : 0, "sum2" : 0, "min" :
Infinity, "max" : -Infinity }, ..., { "offset" : 12, "samples" : 1, "sum" : 74.6, "sum2" : 5565.16, "min" : 74.6, "max" : 74.6 }, ..., { "offset" :
23, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" : "oil_level", "origin" :
ISODate("2015-01-01T00:00:00Z"), "resolution" : "month", "range" : "year", "attrType" : "float" }, "points" : [ { "offset" : 0,
"samples" : 1, "sum" : 0, "sum2" : 0, "min" : 0, "max" : 0 }, ..., { "offset" : 3, "samples" : 0, "sum" : 74.6, "sum2" : 5565.16, "min" :
74.6, "max" : 74.6 }, ..., { "offset" : 11, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName"
: "oil_level", "origin" : ISODate("2015-04-20T12:13:00Z"), "resolution" : "second", "range" : "minute", "attrType" : "float" }, "points"
: [ { "offset" : 0, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" : 22, "samples" : 1, "sum" :
74.6, "sum2" : 5565.16, "min" : 74.6, "max" : 74.6 }, ..., { "offset" : 59, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -
Infinity } ] } { "_id" : { "attrName" : "oil_level", "origin" : ISODate("2015-04-20T12:00:00Z"), "resolution" : "minute", "range" : "hour",
"attrType" : "float" }, "points" : [ { "offset" : 0, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max" : -Infinity }, ..., { "offset" :
13, "samples" : 1, "sum" : 74.6, "sum2" : 5565.16, "min" : 74.6, "max" : 74.6 }, ..., { "offset" : 59, "samples" : 0, "sum" : 0, "sum2" : 0,
"min" : Infinity, "max" : -Infinity } ] } { "_id" : { "attrName" : "oil_level", "origin" : ISODate("2015-04-01T00:00:00Z"), "resolution" :
"day", "range" : "month", "attrType" : "float" }, "points" : [ { "offset" : 1, "samples" : 0, "sum" : 0, "sum2" : 0, "min" : Infinity, "max"
: -Infinity }, ..., { "offset" : 20, "samples" : 1, "sum" : 74.6, "sum2" : 5565.16, "min" : 74.6, "max" : 74.6 }, ..., { "offset" : 31, "samples"
```

NGSICartoDBSink data model

- Data is stored within tables under a user space in carto.com
  - Table names depend on the configured data_model (dm-by-service-path or dm-by-entity) and the working mode (raw historic, raw snapshot or distance historic)

	dm-by-service-path	dm-by-entity
Raw historic	<servicePath>	<servicePath><destination>
Distance historic	<servicePath>+distance	<servicePath><destination>+distance
Raw snapshot	<servicePath>+rawsnapshot	

- Default destination is the concatenation of the notified entityId and entityType

```
<destination>=<entityId>_<entityType>
```

- Custom destinations (an even service paths) can be specified by using the Grouping Rules mechanism
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/grouping_rules_conf/index.html

NGSICartoDBSink table formats

- Supported working modes
  - Raw historic
  - Raw snapshot
  - Distance historic
- *Raw historic* persists a new record for each entity's attribute change, including position
- *Raw snapshot* persist a single record per entity recording an updated version of its attributes, including position
- *Distance historic* persists a new record regarding a set of measures based on the current position and the previous position
  - Max/min stage distance, time and speed
  - Total distance, time and speed
  - Sum and sum2 for distance, time and speed
- All the working modes store the position in the special "the_geom" column

NGSICartoDBSink example

raw historic, dm-by-entity

```
$ curl -G "https://myuser.cartodb.com/api/v2/sql?api_key=xxx" --data-urlencode
"q=SELECT * FROM 4wheels_car1_car"
{"rows":[
   {
      "cartodb_id":1,
      "the_geom":"0101000020E61000006891ED7C3F350BC0D7A3703D0A374440",
      "the_geom_webmercator":"0101000020110F000060EF4D5A961B17C1A59940B...",
      "recvtime":"2016-11-28T08:29:35.44Z”,"fiwareservicepath":"/4wheels",
      "entityid":"car1”,"entitytype":"car”,"speed":112.9,"speed_md":"[]”,
      ”oil_level”:”74.6”,”oil_level_md”:”[]”
   },
   {
      "cartodb_id":2,
      "the_geom":"0101000020E61000009EEFA7C64B370BC03108AC1C5A344440",
      "the_geom_webmercator":"0101000020110F0000A58776A1531D17C1CBEB3D11...",
      "recvtime":"2016-11-28T08:30:35.977Z","fiwareservicepath":”/4wheels",
      "entityid":"car1","entitytype":"car","speed":95,"speed_md":"[]”,
      ”oil_level”:”74.5”,”oil_level_md”:”[]”
   },
   {
      ...
   }]
}
```


FIWARE

NGSICartoDBSink example

distance historic

```
$ curl -G "https://myuser.cartodb.com/api/v2/sql?api_key=xxx" --data-urlencode
"q=SELECT * FROM 4wheels_car1_car_distance"
{"rows":[
   {
      "cartodb_id":1,
      "the_geom":"0101000020E610000080B74082E2470BC05839B4C8762E4440",
      "the_geom_webmercator":"0101000020110F0000F9F1D8616A2B17C18CA42D61...",
      "recvtimems":1480342611051,"fiwareservicepath":"/4wheels",
      "entityid":"car1","entitytype":"car","stagedistance":0,
      "stagetime":0,"stagespeed":0,"sumdistance":0,"sumtime":0,
      "sumspeed":0,"sum2distance":0,"sum2time":0,"sum2speed":0,
      "maxdistance":1.4e-45,"mindistance":3.4028235e+38,"maxtime":1.4e-45,
      "mintime":3.4028235e+38,"maxspeed":1.4e-45,"minspeed":3.4028235e+38,
      "numsamples":1
   },
   {
      "cartodb_id":2,
      "the_geom":"0101000020E61000000F0BB5A679470BC091ED7C3F352E4440",
      "the_geom_webmercator":"0101000020110F000051A09D53112B17C17D410E55...",
      "recvtimems":1480342698000,"fiwareservicepath":"/4wheels",
      "entityid":"car1","entitytype":"car","stagedistance":222.732003988,
      "stagetime":86949,"stagespeed":0.0025616396276898,
      "sumdistance":222.732003988,"sumtime":86949,
      "sumspeed":0.0025616396276898,"sum2distance":49609.5456005104,
      "sum2time":7560128601,"sum2speed":0.00000656199758215071,
      "maxdistance":222.732003988,"mindistance":222.732003988,
      "maxtime":86949,"mintime":86949,"maxspeed":0.0025616396276898,
      "minspeed":0.0025616396276898,"numsamples":2
   },
   {...}]
}
```

Further documentation

- You may find all the details regarding all available sinks here:
  - HDFS: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_hdfs_sink
  - MySQL: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_mysql_sink
  - CKAN: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_ckan_sink
  - PostgreSQL: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_postgresql_sink
  - MongoDB: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_mongo_sink
  - STH: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_sth_sink
  - Kafka: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_kafka_sink
  - DynamoDB: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_dynamodb_sink
  - CartoDB: http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_cartodb_sink

Advanced general features

- Batching mechanism
- Fault tolerance
- Grouping rules
- Name mappings
- Round-Robin channel selection
- Kerberos authentication for HDFS
- Multi-instance support
- Management API
- New encoding

Batching mechanism

- Let’s assume 100 NGSI notifications to be persisted
- Among them, there are 3 entities involved
  - That means 3 HDFS files will be created and updated
- Until now, per each notification a write was done
  - 100 writes!
- If using batching, the notifications are accumulated per entity until certain batch size is reached, or certain timeout is reached. Then, per-entity data is aggregated and a single per-entity write is done
  - 3 writes!
- In the worst case, i.e. the 100 notifications belong to 100 different entities, Cygnus will behave as usual

How to use batching

- Two new parameters are added to the sink configuration:

```
cygnus.sinks.hdfs-sink.batch_size=100
cygnus.sinks.hdfs-sink.batch_timeout=20
```

- Batching parameterization must be carefully studied, based on:
  - The number of entities, i.e. configuring a batch size of 100 has no sense if Orion notifies a sequence of updates about 200 different entities
  - The notification frequency, i.e. configuring a batch timeout of 10 has no sense if the notification frequency is 15
  - The sink, e.g. maximum batch size allowed by DynamoDB API is 25
  - The channel transaction capacity, i.e. Configuring a batch size of 2000 has no sense if the channel the sink reads only allows for reading 1000 notifications per transaction

Fault tolerance

- As seen, Flume agents contain internal channels communicating sources and sinks
- Such channels accomplish an additional purpose:
  - To absorbe peaks of notifications
- In addition, Cygnus adds another internal queue:
  - To absorbe not persisted batches of notifications
  - It is batch-based, not notification-based
  - Batches within this queue have a TTL, which decreases on attemps
  - Additionaly, a list of retry intervals is configured
- Batch TTL and retry intervals parameterization:

```
cygnus.sinks.hdfs-sink.batch_ttl=10
cygnus.sinks.hdfs-sink.batch_retry_intervals=5000,10000
```

Grouping rules

- **Default destination** (HDFS file, mMySQL table or CKAN resource) is obtained as a concatenation:

```
<destination>=<entityId>_<entityType>
```

- It is possible to **group different context data** thanks to this regex-based feature implemented as a Flume interceptor:

```
cygnusagent.sources.mysource.interceptors=gi
cygnusagent.sources.mysource.interceptors.gi.type=com.iot.telefonica.cygnus.interceptors.NGSIGroupingInterceptor$Builder
cygnusagent.sources.mysource.interceptors.gi.grouping_rules_conf_file=/usr/cygnus/conf/grouping_rules.conf
```

- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/grouping_rules_conf/index.html

Grouping rules syntax

```
$ cat /usr/cygnus/conf/grouping_rules.conf
{
    "grouping_rules": [
        {
            "id": 1,
            "fields": [
                "entityId”,
                "entityType"
            ],
            "regex": "Room\\.(\\d*)Room",
            "destination": "numeric_rooms",
            "fiware_service_path": "rooms"
        },
        {
            "id": 2,
            "fields": [
                "entityType"
            ],
            "regex": "Room",
            "destination": "other_rooms",
            "fiware_service_path": "rooms"
        }
    ]
}
```

Name mappings

- It is an **evolution** of grouping rules
- Not only the contatenation of entity ID and entity type and the FIWARE service path can be changed, but:
  - FIWARE service - Entity type
  - FIWARE service path - Attribute name
  - Entity ID - Attribute type
- It works as another custom Flume interceptor:

```
cygnusagent.sources.mysource.interceptors=nmi
cygnusagent.sources.mysource.interceptors.nmi.type=
com.iot.telefonica.cygnus.interceptors.NGSINameMappingsInterceptor$Builder
cygnusagent.sources.mysource.interceptors.nmi.
name_mappings_conf_file=/usr/cygnus/conf/name_mappings.conf
```

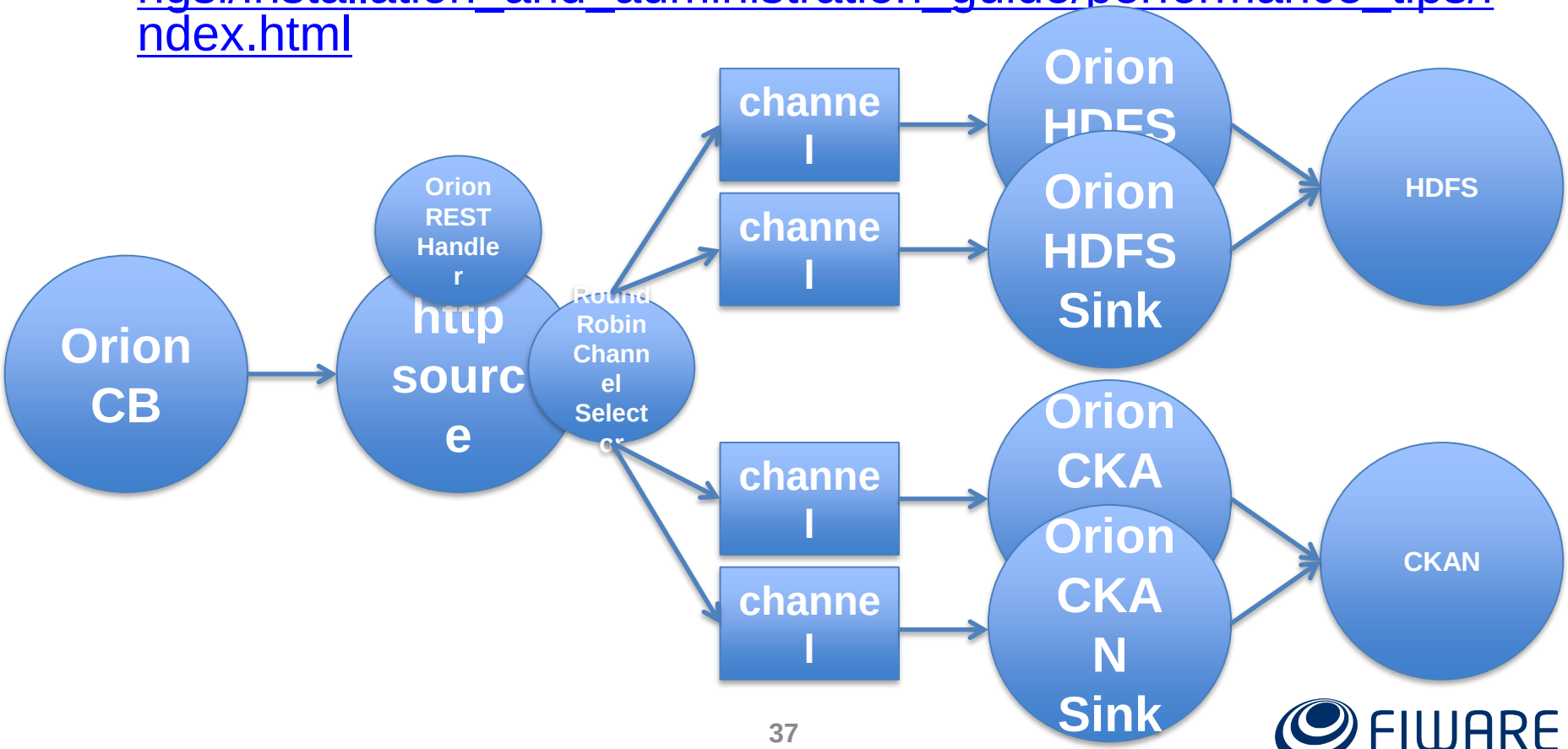
- Grouping rules should be **deprecated** in favour of name mappings
- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/name_mappings/index.html

Name mappings syntax

```
$ cat /usr/cygnus/conf/name_mappings.conf
{
  "serviceMappings": [
    { "originalService": "frb”,
      "newService": "new_frb”,
      "servicePathMappings": [
        { "originalServicePath": "/any”,
          "newServicePath": "/new_any”,
          "entityMappings": [
            { "originalEntityId": "Room1”,
              "originalEntityType": "Room”,
              "newEntityId": "new_room1”,
              "newEntityType": "new_room”,
              "attributeMappings": [
                { "originalAttributeName": "temperature”,
                  "originalAttributeType": "centigrade”,
                  "newAttributeName": "new_temp”,
                  "newAttributeType": "new_cent”
                },
                ...]
            },
            ...]
        },
        ...]
    },
    ...]
}
```

Round Robin channel selection

- It is possible to configure **more than one channel-sink pair** for each storage, in order to increase the performance
- A custom **ChannelSelector** is needed
- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/performance_tips/index.html



RoundRobinChannelSelector configuration

```
cygnusagent.sources=mysource
cygnusagent.sinks=mysink1 mysink2 mysink3 mysink4
cygnusagent.channels=mychannel1 mychannel2 mychannel3
mychannel4

cygnusagent.sources.mysource.type=...
cygnusagent.sources.mysource.channels=mychannel1
mychannel2 mychannel3 mychannel4
cygnusagent.sources.mysource.selector.type=com.telefo
nica.iot.cygnus.channelselectors.RoundRobinChannelSel
ector
cygnusagent.sources.mysource.selector.storages=2
cygnusagent.sources.mysource.selector.storages.storag
e1=mychannel1 mychannel2
cygnusagent.sources.mysource.selector.storages.storag
e2=mychannel3 mychannel4
```

Kerberos authentication for HDFS

- HDFS may be secured with **Kerberos** for authentication purposes
- Cygnus is able to **persist on *kerberized* HDFS** if the configured HDFS user has a registered Kerberos principal and this configuration is added:

```
cygnusagent.sinks.hdfs-sink.krb5_auth=true
cygnusagent.sinks.hdfs-sink.krb5_auth.krb5_user=username
cygnusagent.sinks.hdfs-
sink.krb5_auth.krb5_password=xxxxxxxxxxxx
cygnusagent.sinks.hdfs-
sink.krb5_auth.krb5_login_file=/usr/cygnus/conf/krb5_login.c
onf
cygnusagent.sinks.hdfs-
sink.krb5_auth.krb5_conf_file=/usr/cygnus/conf/krb5.conf
```

- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/ngsi_hdfs_sink/index.html#section3.3

Multi-instance support

- When running as a **process**, the only configuration file needed by a Cygnus agent is its **agent configuration file**
- When running as a **service**, in addition to the agent configuration file it is required **an instance configuration file**
- Cygnus service will **run a Cygnus process per each pair** of instance-agent configuration files
- Instance configuration file looks like:

```
CYGNUS_USER=cygnus
CONFIG_FOLDER=/usr/cygnus/conf
CONFIG_FILE=/usr/cygnus/conf/agent_<id>.conf
AGENT_NAME=cygnusagent
LOGFILE_NAME=cygnus.log
ADMIN_PORT=8081
POLLING_INTERVAL=30
```

- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-common/installation_and_administration_guide/cygnus_agent_conf/index.html

Management API

- REST API for management purposes
- Categories:
  - Statistics API
  - Grouping Rules API
  - NGSI Subscriptions API
  - Admin API
  - Admin Log4j Loggers API
  - Admin Log4j Appenders API
- http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-common/installation_and_administration_guide/management_interface/index.html

New encoding, why?

- Old plain encoding has flaws:
  - Not accepted characters are replaced with underscore
  - The concatenator character is underscore as well
  - The more restrictive storage encoding constraints are “exported” to the other storages
  - Except for HDFS and STH, the slash within FIWARE service paths is discarded
- Example. These 3 different entites, once encoded, result in the same HDFS file
  - ID: Room$1, type: Room, HDFS file: Room_1_Room
  - ID: Room_1, type: Room, HDFS file: Room_1_Room
  - ID: Room, type 1_Room, HDFS file: Room_1_Room
- New encoding is required
  - Per character specific
  - Per storage specific
  - Concatenator different than underscore (which is now another simple character)

New encoding, how?

- Solution
  - Encode each character, when require by the storage, with x<UNICODE>
  - Use xffff as concatenator
  - If a user defined string matches this encoding format, it is encoded as xx<UNICODE>
  - https://en.wikipedia.org/wiki/List_of_Unicode_characters
- Example:
  - ID: Room$1, type: Room, HDFS file: Roomx00241xffffRoom
  - ID: Room_1, type: Room, HDFS file: Room_1xffffRoom
  - ID: Room, type 1_Room, HDFS file: Roomxffff1_Room
- Now the slash within FIWARE service paths is encoded as part of the service path value
  - /gardens → x002fgardens
- Per storage specific encdoing rules:
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/flume_extensions_catalogue/introduction/index.html

How to get Cygnus

- From sources
  - https://github.com/telefonicaid/fiware-cygnus
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/install_from_sources/index.html
- From FIWARE RPM repository
  - http://repositories.testbed.fiware.org/repo/rpm/6/x86_64/
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/install_with_rpm/index.html

```
$ yum install cygnus-common
$ yum install cygnus-ngsi
```

- From Dockerhub
  - https://hub.docker.com/r/fiware/cygnus-ngsi/
  - http://fiware-cygnus.readthedocs.io/en/1.6.0/cygnus-ngsi/installation_and_administration_guide/install_with_docker/index.html

```
$ docker pull fiware/cygnus-ngsi
```

Thank you!

http://fiware.org
Follow @FIWARE on Twitter