COSMOS

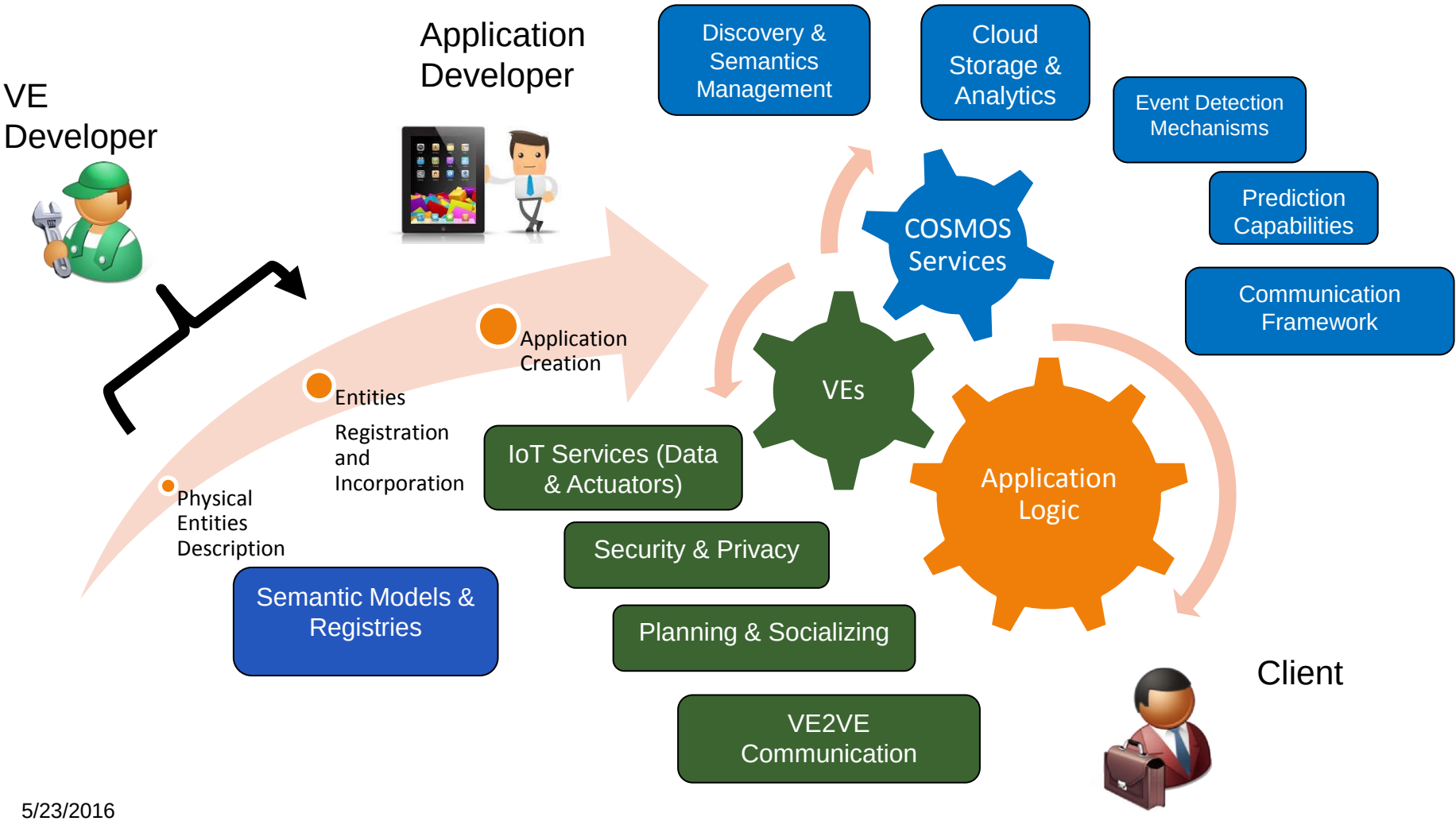
Cultivate Resilient Smart Objects for Sustainable City Applications

1st IoT Challenge/Workshop

NTUA

23/5/2016

The COSMOS Story & Phases (Y1)



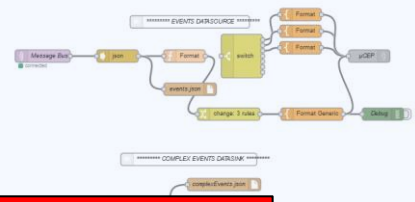
COSMOS VE side apps

- VE= Virtual Entity
  - The digital representation of a physical entity, in this case the house
- Typically residing on a gateway type of hardware
  - E.g. Raspberry Pi
- Implementing a specific functionality with relation to Smart Home operation
- Potentially using a number of platform or external based services
  - E.g. Data management, Data Analytics, Event generation, Weather services etc.
- Reasoning on situations and providing solutions that are turned into actuations

COSMOS VE side apps

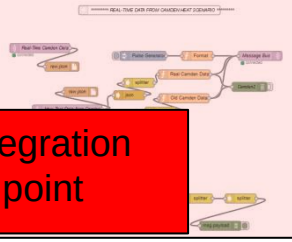
Bluetooth node

Generate VE Events



Integration point with SE flow

Data Input from Smart Home



Integration point

Data Output- To other Ves or platform

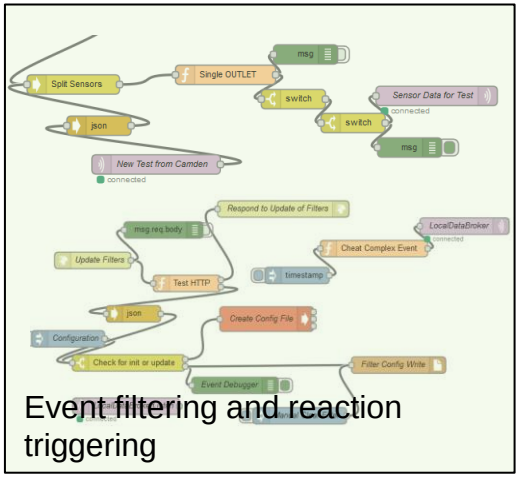
Privelets

Planner

XP Sharing

Social Monitoring

T&R Model



Event filtering and reaction triggering

Runtime Management

Actuate

Events on top of events

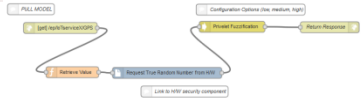
Scenario A

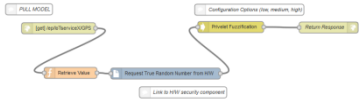
- **Scenario A** refers to the creation of an application for assisting Smart Home users with hearing impairment. The purpose of the app is:
- to capture and identify environment sounds, analyze and understand their purpose and notify in a different manner the person involved
- Potential notifications may be of a variety of actions, such as a smart watch notification, a visual notification (e.g. light bulb blinking etc.).
- Participants are expected to utilize test sounds and apply the analysis of their choice (initial suggestions given) and investigate potential matchmaking and generalization aspects for sound categorization.

Scenario A Challenges

- Use as input sound files, use certain analysis tools to extract the appropriate parameters (e.g. dominant frequencies) and format them in a way that they can be used as input to the Case-based Reasoner provided by the COSMOS project (see “available tools” for details).
- Test similar and dissimilar sounds to identify whether the selected parameters and thresholds are the appropriate ones.
- Implement an actuation/notification scenario.
- Propose other functionalities that could be of interest for the specific application.

Scenario A Process



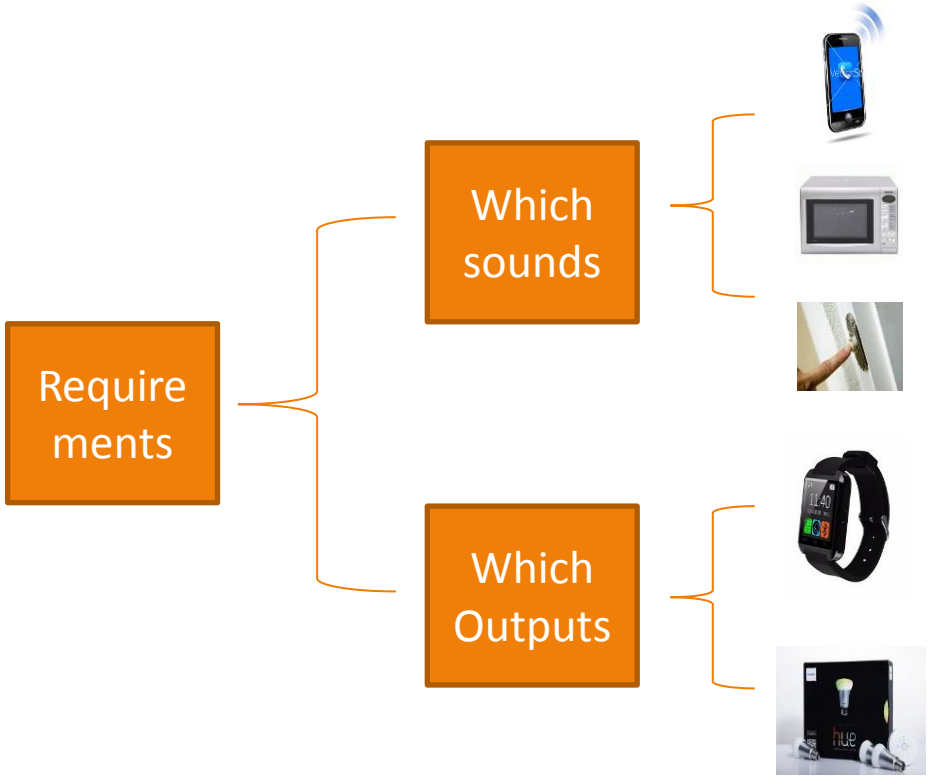


Apply Actuation

CBR/Planner Categorization

General Process

- Requirements List
  - Inputs, outputs and delays
- Find Test sounds (wav)
  - www.soundjay.com
- Sound source (with microphone or local test file)
- Check potential modifications needed (e.g. in format from wav to another format needed)
- Analyze through a tool of your choice
- Define CBR structure and format
  - Decide on metrics and related to the tool you will use in the previous step
- Define and implement actuation plan
- Implement via Node-RED (preferably) or otherwise



Example tools you can use for sound analysis

- https://blog.mornati.net/raspberrypi-motion-and-noise-detection/
- Has some nice statistics on noises (check next slides), may be useful for vector
- Install ruby on system
- sudo ./noise_detection.rb –d
  - Get soundcard id
- Need to remove the –D option from the arecord command
- Need to have created the /etc/noised directory beforehand
- Test
- Sudo ./noise_detection.rb -t SOUNDCARD_ID -v
- Run:
- sudo ./noise_detection.rb -m 0 -n 0.30 -e test@mail.com –v

- Uses SOX linux utility which is also available in Raspberry distribution

- Many metrics from its statistics depend on amplitude so probably not the best for categorization

- Maybe SOX has also other capabilities

Example Stats from SOX

- Maximum amplitude: 0.367310
- Minimum amplitude: -0.424530
- Midline amplitude: -0.028610
- Mean norm: 0.021940
- Mean amplitude: 0.000001
- RMS amplitude: 0.043971
- Maximum delta: 0.566772
- Minimum delta: 0.000000
- Mean delta: 0.012360
- RMS delta: 0.030856
- Rough frequency: 893
- Volume adjustment: 2.356

Other more standard options

- Use Fast Fourier Transform to find frequencies
  - And check e.g. X more dominant frequencies for each sound
- Functions exist in Node.js
  - fft-js
  - DSP.js

Scenario B

- **Scenario B:** Scenario B is more generic and refers to any kind of meaningful combination of functionalities that may be achieved, in the context of a Smart home or Smart city application. E.g. :
- Registration and usage of Social data (e.g. from Twitter), their aggregated (e.g. in order to identify areas of happiness) or individualized analysis (e.g. based on sentiment identification)
- Feed of sensor data with usage of privacy filters and the potential combination of information in order to generate meaningful events that are of interest.
- Exploitation of Open Smart city data available in the internet

Scenario B Challenges

- Integrate at least one external data source (example given on Smart City open data).
- Integrate at least one data management solution. You can use personal data that can be fuzzified through the Privelets mechanism (see “available tools” for details).
- Apply at least one filtering/processing layer (e.g. use Sentiment Analysis Toolkit which is provided by LeanbigData FP7 Project).
- The innovative nature and originality of the proposed combinations will be taken under consideration.

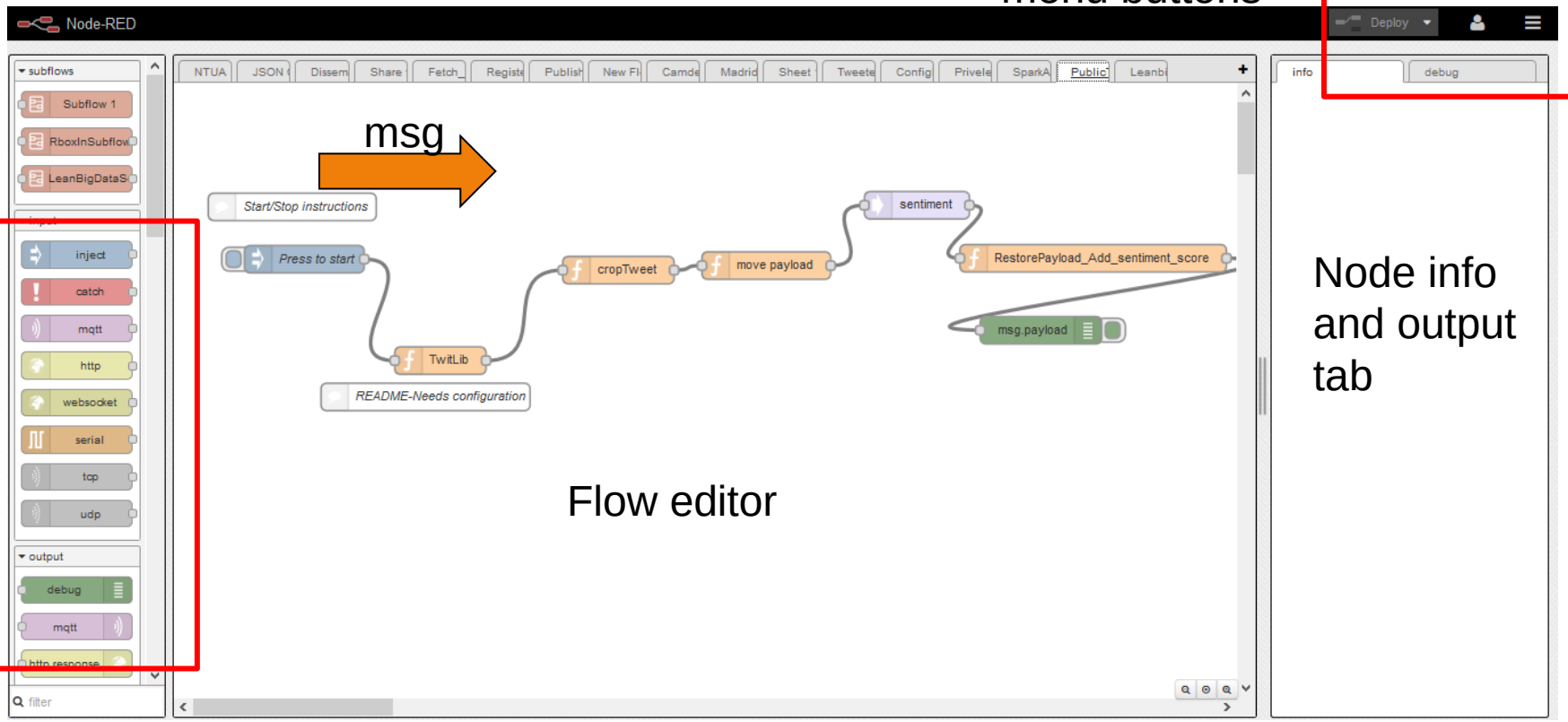
Node-RED

- Visual tool for creating workflows, integrating different systems and implementing a specific logic
- Consists of nodes that can be instantiated in a given flow
- Asynchronous communication of **messages**
- Message is passed from one node to another
- Each node uses fields of the message for some purpose (e.g. arguments) and adds/alters other fields. The changed message is then passed on to the next node(s)
  - E.g. msg.payload.state="good";
- Based on the Node.js framework


```
{
  "payload":{
    "state":"Bad",
    "status":"0",
    "img":"http://informo.munimadrid.es/informo/Camaras/Cama
    "lon":"-3.7362283",
    "lat":"40.4339686",
    "tf":"16:40:17",
    "ts":"1445438417305",
    "DiffIntensity":"0.000000",
    "DiffSpeed":"0.000000",
    "ValueIntensity":"1200",
    "ValueSpeed":"47",
    "codigo":"PM12061"
  },
  "_msgid":"66483e08.99b7c",
  "state":"Bad",
  "topic":"trafficEvent"
}
```

Node-RED GUI

Deployment and menu buttons



Node info and output tab

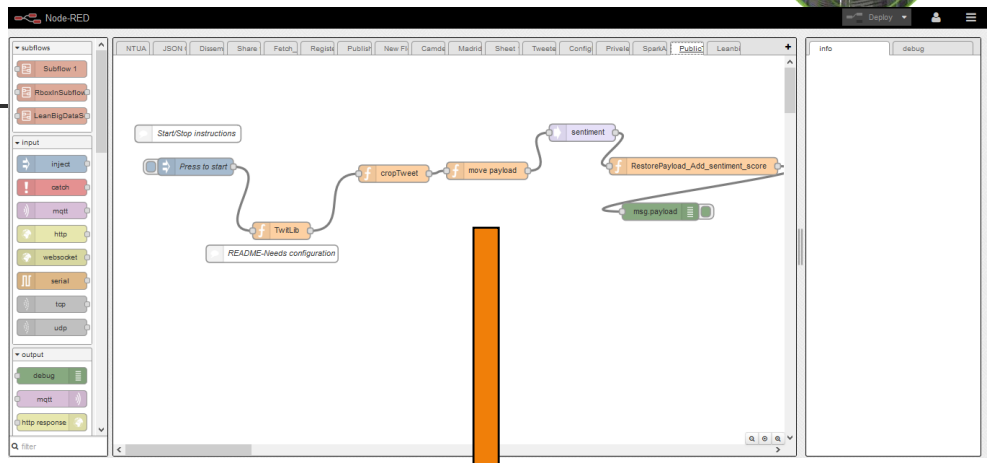
Flow editor

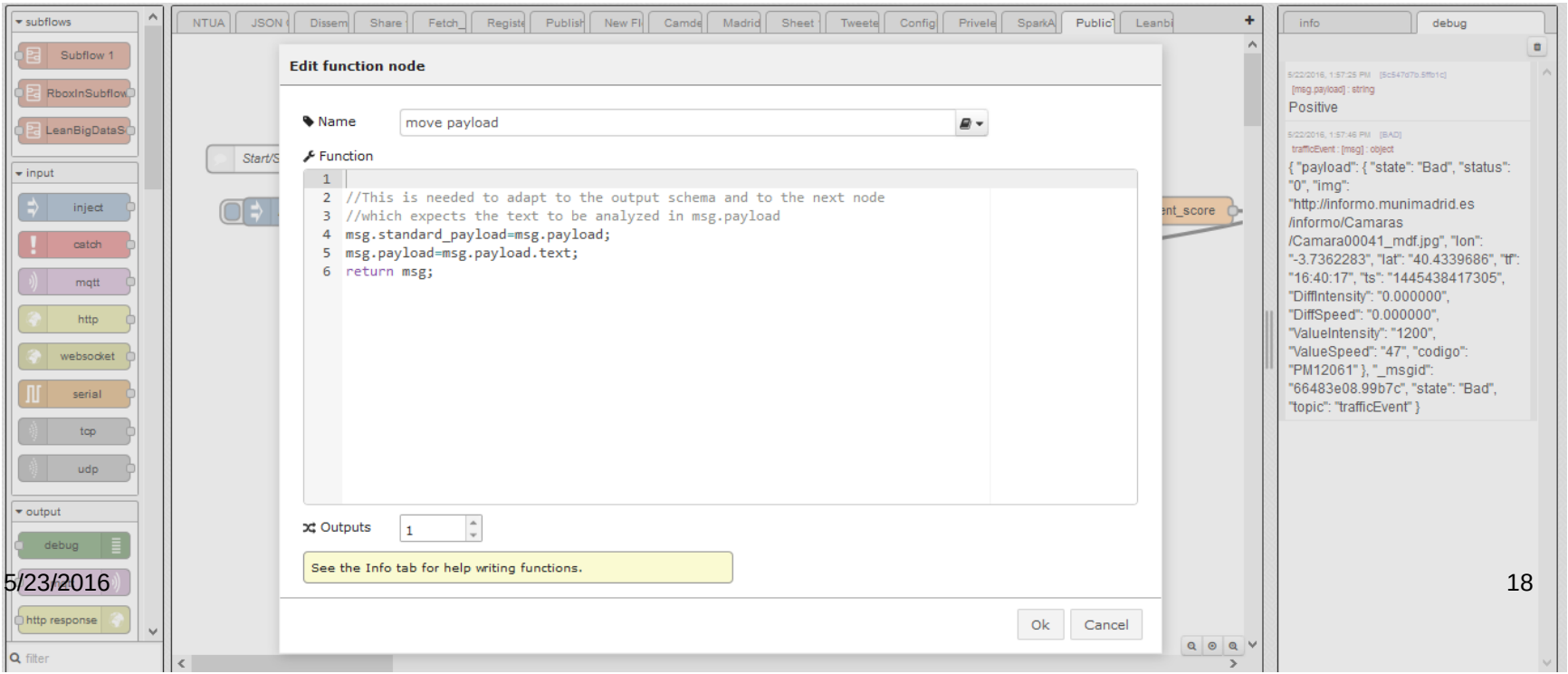
Node pallete

Node-RED built-in nodes

- Typical functionalities such as
  - Create service
  - Call service
  - Publish/Subscribe to an external messaging system (mqtt)
    - Not implementing the messaging system!
  - Write to file, monitor file, change format or object type (from string to JSON object)
    - Beware of what type is passed!!!!!! If for example {\" then this is a string, it will be an error when accessing msg.field
  - Perform a Tweet
  - Etc.
- Custom javascript functions (very helpful!)
  - Helps you manipulate a message, change its fields or perform any other operation not available as a node
  - Msg is received in input, changed in content and then forwarded to output (with return msg;)
- You can also remain in a node of the flow (e.g. listening on an external event to arrive) while sending the message to the next ones
  - Node.send(msg) in the function

Custom function



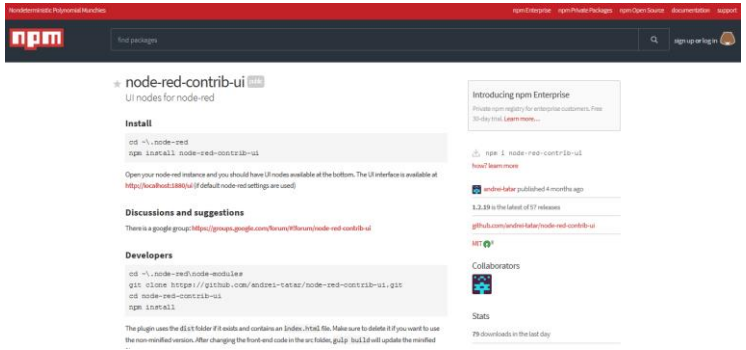


Node-RED Flow deployment

- Runs as a server. Once you have created a flow, you can deploy it in order to start being operational
- Always use Deploy modified flows
  - Option in top right corner in Deploy button
- This ensures that any initialization actions in some of the flows are not needed each time you change and redeploy another flow

Node-RED External Nodes

- Add external node-RED nodes. A large variety of available implementations exist, with client nodes for e.g.
  - DB systems (mongoDB, mysql etc.)
  - Interesting APIs (Google Maps, Calendar etc.)
  - Easy to install through npm repository
    - Typically a sudo npm install –g <package_name>
  - Once installed they appear in the left side panel (with a refresh and node-red restart)
  - Typically included in www.npmjs.com with a name of nodered-contrib-XXX
- You may also create your own node or flow and publish it on the repositories
  - Check instructions on nodered.org!



Node-RED import of Node.js functions (SOS)

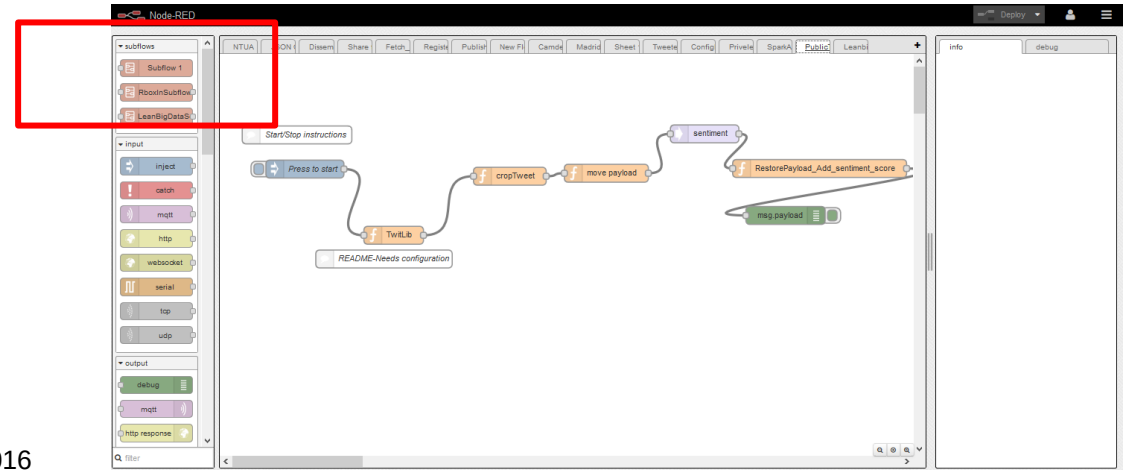
- Numerous node.js functions exist that have not been packaged as Node-RED nodes and thus can not be directly installed

- However they can be imported with some manual steps
  - It is worth it!
  - Check simplified instructions in https://github.com/COSMOSFP7/COSMOS-Platform-side/blob/master/AddNodejsInNodered.pdf

Node-RED Subflows

- Ability to group flows into one node, defining the input and output of the overall flow (and documenting through comment nodes)
- Helps you simplify a diagram and reuse parts easily

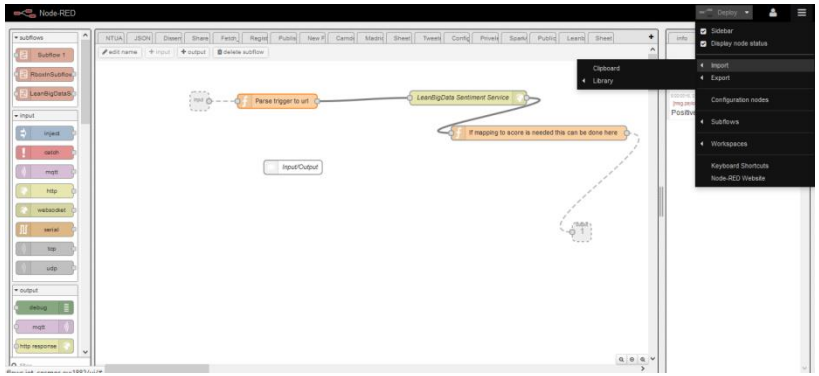
Your created subflows



Node-RED Generic Available tools

- Flows available from https://github.com/COSMOSFP7/

- In order to use them, you can even browse through the folders online, copy the *.json file contents and then go to Node-RED-> top right menu item-> Import -> From Clipboard and paste the json text in the box



Importing a flow

- If some flow consists of a non-builtin node that you have not installed, you will get an error message
- If you install it it will be ok
- Some flows may need manual configuration (typically mentioned in the info button or in comment tabs in the flow)
  - E.g. paths to disk
  - URLs
  - etc

Node-RED Generic Available tools

- Platform side flows
  - Event correlation examples (Application Layer)
  - Data ingestion examples (Madrid Traffic and Twitter)
    - You need twitter account and developer credentials for this
  - Usage of sentiment analysis
  - Usage of local command line tools (such as Apache spark) via Node-RED
- VE side flows
  - Planner code and flows
  - Privelets flows

Flow Demos

- Copy from github
- In a tab in flows

Link to competition website

- http://www.iot-cosmos.eu/node/1998

Evaluation

- Please evaluate tools and processes in the following form!:

- https://docs.google.com/forms/d/1a66UtJjF5Nl9L-ab1jYu6ijGNkau5yoOHzPimzqlY9c/viewform