Creation of a new event by a producer

In order for a producer to create a new event, the following steps need to be undertaken:

- Create an account on Eventflows.com
- Check out flows from COSMOS that may assist you to get started
    - https://github.com/COSMOSFP7/COSMOS-Platform-side
    - More details on this step are included in Section 0
- Imagine a combination of data or data&processing that can help identify an interesting event and create the sequence that generates it. This is probably the most critical step in order for the event to be meaningful and interesting for external users. Especially events that may be used in the context of mobile apps should be considered, since these cases would have a wide exploitation field. For the sequence, any kind of technology may be utilized, however COSMOS tools are mainly based on Java and Node-RED. Especially for the latter it is strongly advised to utilize it since it is very flexible and able to apply data transformations easily and combinations of published information
- Register your event in the marketplace and insert the necessary descriptions (relevant description follows)
    - E.g. data schema of messages
- Credentials for accessing the Messaging system (AMQP protocol) will be sent via email along with the endpoint to which the data should be pushed
- Use and configure provided Node-RED flows or Java clients for pushing data to the provided endpoint (dedicated Section follows)
    - https://github.com/COSMOSFP7/Eventflows-Marketplace/tree/master/NodeRED_Clients

Usage of existing flows as starting point

One of the Marketplace benefits is to abstract large parts of the process for an event generation, through the reutilization of common flows. For this reason a number of flows have been made available through our GitHub repository, that can be easily copied and integrated directly in any new attempt for event creation. Details on how to do this are included in the following paragraphs, using as an example the Twitter registration flow provided.

The detailed import steps are (applies to all flows):

- Data flow for registering to Twitter and getting data. The flow is available here:
    - https://github.com/COSMOSFP7/COSMOS-Platform-side/tree/master/DataIngestion/TwitterDataIngestion
    - The flow registers to Twitter API and receives tweets based on geolocation.
        - Can be manipulated afterwards in multiple ways
            - Stored e.g. in a DB , object storage etc. and analyzed statistically to find peaks
            - Passed through sentiment analysis to find happy and sad spots
- In order to use it, copy the json file and paste it in a Node-RED tab (Figure 1)
    - Import-> From Clipboard
    - Paste json file
    - Flow will appear in Tab (can be connected to whatever output e.g. MQTT node etc.)
- Needs configuration. Details are included as comments in the flow

- Credentials as a Twitter registered developer
- Configuration of geographical coordinates if one is interested in another area other than Madrid

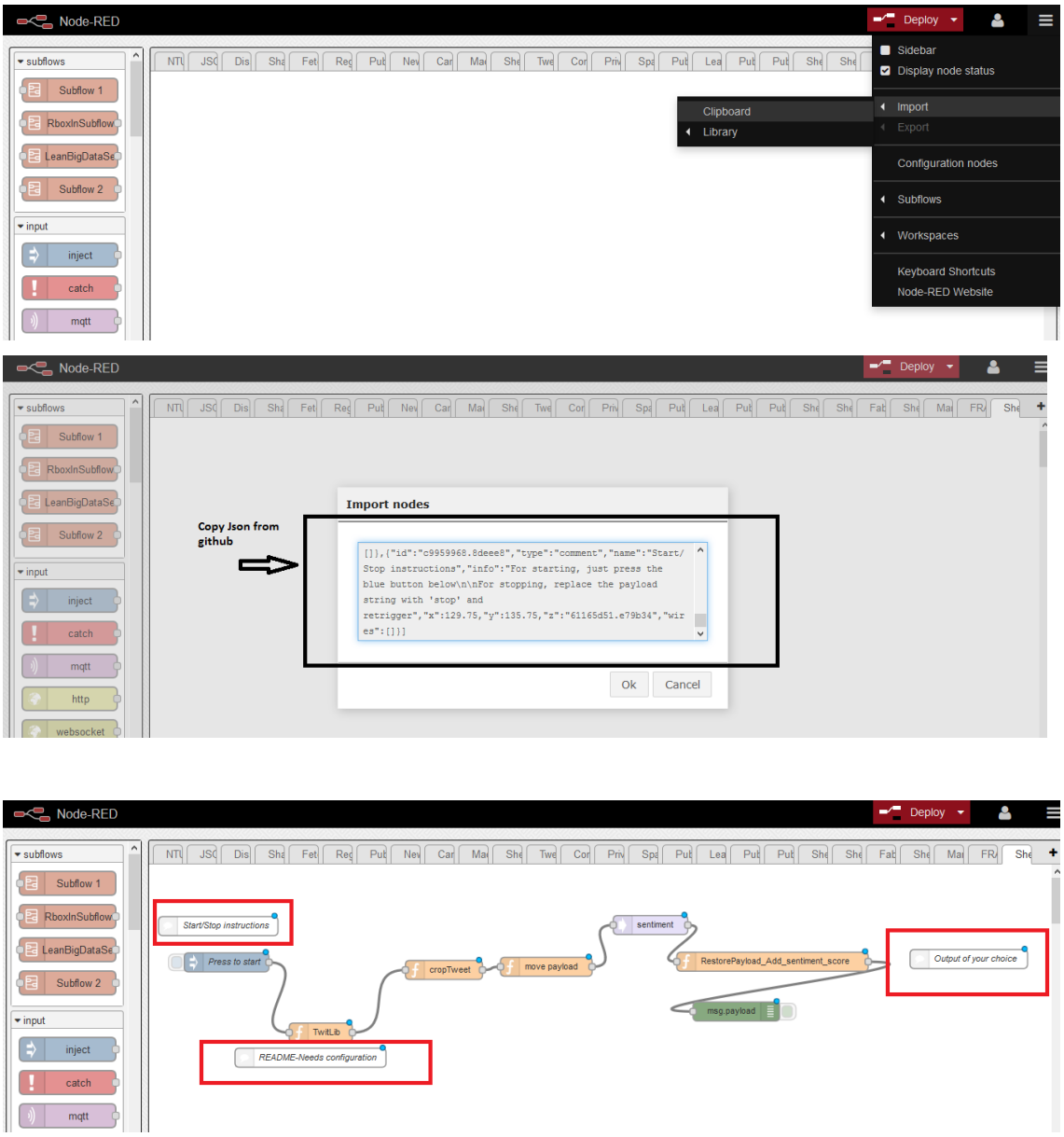

Figure 1: Process of importing an existing Node-RED flow and configuration details

Description of an event on the Marketplace

Following an account creation on the Marketplace and an event production, the producer needs to register this event on the Marketplace. In order to do that, they need to:

- Follow the process of posting a new listing (button available on Front end)

- Define the category in which the listing is included for semantic and filtering purposes (indicative categories may include City event, Weather event, Social networks event etc.)
- Define the type of the listing (free or for purchase)
- Populate the fields on the event template description that appears in Figure 2

Especially for the latter, the fields are the following:

- Description of the event. This ideally should include details on how it is produced, as well as key information with relation to e.g. boundaries of the event or of the provided fields.
- JSON schema. This should include the schema under which the event is published. There is no limitation with relation to the content of the schema, since this is event-related, only that it should be in JSON.
- JSON example. An example data item based on the previous schema could be useful for potential consumers.
- Estimated produced number of events per hour. This is an indicative field that should be populated if applicable, in order to inform potential consumers regarding the message rate they are expected to ingest. Remember that consumers also login through the messaging system and receive the data, but the post processing of this data on their end should also be present. Therefore this information will be useful in order for them to determine what kind of resources they need to accommodate for this purpose.
- QoS Features. This is an indicative field that may be populated by the producer in order to let consumers know key Quality of Service characteristics of the event. Examples of such cases may be the error of a prediction service, as tested against specific conditions/data set, the sensitivity of the sensors, the rate of guaranteed data acquisition etc.

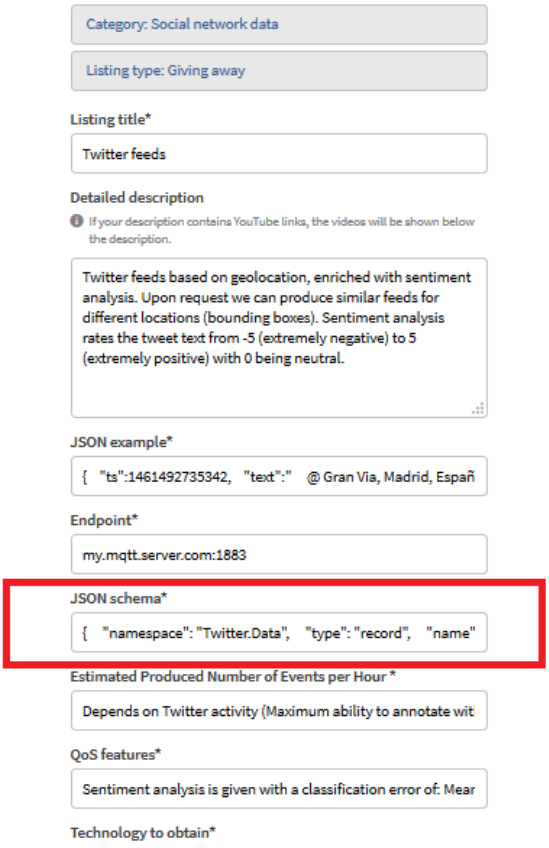

Figure 2: Event template description on Eventflows.com

Pushing to the Marketplace Messaging System

After producing the event and preparing its description for the Marketplace, now it is time for actually pushing it for publication in the Eventflows messaging system. As mentioned in the previous sections, the users of the Marketplace (both consumers and producers) are expected to use the common Pub/Sub system based on RabbitMQ in order to exchange information. However, in order for the system to have consistency in terms of namespaces and management, these users have limited rights for configuration. This creates the problem that widely used Node-RED nodes that could be used for accessing RabbitMQ (based on the AMQP protocol) like https://www.npmjs.com/package/node-red-contrib-amqp can not be used. The reason for this is that the specific implementation requires extensive rights in order to be able to operate, such as configuration rights for creation of exchanges and queues. However in the marketplace pub/sub system, these rights are reserved only for the administrator role and not for the producer or consumer role.

For this reason we created a set of customized Node-RED flows that may be able to override this issue. The main library used (from many tested) was amqplib (https://github.com/squaremo/amqp.node), which was included and configured appropriately in the subscriber and consumer implementation flows (and alternatively Java clients also) that have been published on the COSMOS GitHub (Figure 3). The flow consists of a main function incorporating the node.js function of amqplib, appropriately configured to be used with the Eventflows messaging system. However a set of configuration steps are also needed, as indicated in the README file included in the flow, which includes inserting e.g. credentials acquired, exchange name and url of the messaging system, as provided by the marketplace. Furthermore, a manual testing trigger is included as well as a json node which may be needed if the produced event is encoded as a JSON object. In that case it needs to be transformed to a string in order to be visible to the consumer. The link of the produced event to the publication flow is expected to be performed either in this JSON node or in the function node ("Push data to marketplace").

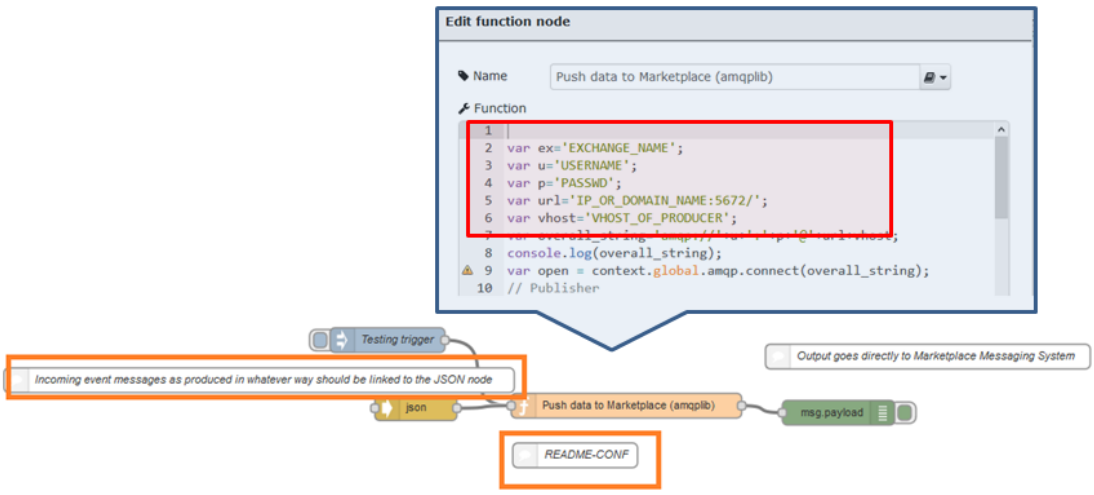

Figure 3: Producer events publication flow