

Node-RED, ServloTicy and InTeGreen

Matthias Dieter Wallnöfer, TIS innovation park, Bolzano/Bozen - Italy

[Overview](#)

[Installation](#)

[ServloTicy \(Glue-Things, Compose\)](#)

[Plugins \(Nodes\)](#)

[ServloTicy](#)

[InTeGreen](#)

[Deployment/duplication of pre-existing flows](#)

Overview

- <http://nodered.org/>
- Article: <http://www.techrepublic.com/article/node-red/>
- Video: <https://www.youtube.com/watch?v=f5o4tlz2Zzc>
- based on Node.js
- is kind of an orchestrator
- the plugins/components are called “nodes”
- based on *sources* (inject/input node) and *sinks* (output node)
- allows the exchange and adaption (function nodes) of data between different protocols/software
- the diagrams are called *data flows* and are stored as JSON

Installation

- Install/upgrade Node.js
 - on Ubuntu 12.04 Precise: the manual update from rel. 0.6 to rel. 0.10 was necessary
 - on Fedora 19 the shipped version was okay
 - on Raspbian (Raspberry Pi) perform these 2 commands in order to install a recent Node.js:
`wget http://node-arm.herokuapp.com/node\_latest\_armhf.deb`
`sudo dpkg -i node_latest_armhf.deb`
- Guide: <http://nodered.org/docs/getting-started/installation.html>
- A note for Raspberry Pi and other remote SSH installations: use `wget` to download archives since this saves you from downloading it locally and copying with `scp`. Example:
`wget https://github.com/node-red/node-red/archive/0.8.1.zip`
- Download link here: <http://nodered.org/>, which provides you with a ZIP archive
Also the GIT checkout is possible, but the directory name changes (*node-red*)
- Extract the Node-RED ZIP archive and perform the installation:
`unzip node-red-0.8.1.zip`

cd node-red-0.8.1

npm install --production

- Download additional nodes, since the basic version contains only few ones:
 - In the Node-RED directory:
cd nodes
 - 3 interesting Node-RED node repos exist:
 - Matthias Wallnöfer's repo (<https://github.com/mwallnoefer/node-red-nodes/tree/integreen>) with both InTeGreen and ServloTicy support
 - Charalampos Doukas' repo (<https://github.com/hdoukas/node-red-nodes>) with ServloTicy support
 - Official default repo (<https://github.com/node-red/node-red-nodes>)
 - Please clone Matthias' nodes repo to have both InTeGreen and ServloTicy support. Otherwise you may download it as a ZIP archive:
wget https://github.com/mwallnoefer/node-red-nodes/archive/integreen.zip
unzip integreen.zip
 - For Charalampos' repo the download commands would be:
wget https://github.com/hdoukas/node-red-nodes/archive/master.zip
unzip master.zip
 - For the official default repo the download commands would be:
wget https://github.com/node-red/node-red-nodes/archive/master.zip
unzip master.zip
- When you plan to use database systems like MySQL, MongoDB or access hardware like Arduino over the serial port, you may need additional Node.js modules.
 - Run Node-RED in verbose mode to find all missing Node.js modules
node red.js -v
 - Stop it by pressing Ctrl+C and install the desired modules by *npm*, eg. for accessing the serial port on Arduino:
cd ~
npm install serialport
 - A more concrete example when you intend to use MySQL, MongoDB, SQLite3, the serial port, Arduino, Wake-on-LAN and Philips Hue bulbs could be:
cd ~
npm install mysql mongodb sqlite3 serialport arduino-firmata wake_on_lan node-hue-api
- Run it (stopping is done with Ctrl+C):
node red.js
on Raspberry Pi set also the memory limitation:
node --max-old-space-size=128 red.js
- Now Node-RED should be available in the web-browser at <http://localhost:1880>
- The software is client-server-based, so it is no problem having it running on remote machines. Be aware that Node-RED could crash under certain conditions, so make sure that you are able to restart it then.

ServIoTicy (Glue-Things, Compose)

- ServIoTicy (or Compose) is a very powerful Internet of Things (IoT) platform. It can be used as things' data storage but also for data elaboration. Please find more information on <http://www.servioticy.com/>, <http://www.gluethings.com> and <http://www.compose-project.eu/>.
- When you plan to use it you need a valid account first
- Please look at http://iotogether.compose-project.eu/?page_id=122 for a how-to for using Glue-Things (= public ServIoTicy-Compose cloud) and Spark, if you plan to use it.
- Register on <http://www.gluethings.com/platform/smart-object-manager/>
- When you have logged in, create a first service object (SO) with "Sensor Type" "Custom template". This one is intended testwise and may be called "Test".
- On the following screen you will notice the **API token**, which is basically your password
- The access URLs are then as follows: <http://api.servioticy.com/>..., eg. for getting SOs streams:
<http://api.servioticy.com/1410877270391637269ea1ddf4cc4879ae0cf783a6296/streams/>
- You will need this information when accessing ServIoTicy with both Node-RED or curl on the command line

Plugins (Nodes)

- **The Node-RED ServIoTicy node** has not yet been incorporated in Node-RED's official nodes repository.

There was already a freely available node (plugin) for COMPOSE in Node-RED.

Originally written by Charalampos Doukas, the author improved its stability and versatility. In the beginning it could only access the public COMPOSE/ServIoTicy cloud and all the parameters needed to be set at configuration time. The author changed this to permit accesses also to other (private) ServIoTicy instances and enabled the parameter setting on run-time over message parameters.

Alternatively users could also access the ServIoTicy API directly by using the HTTP node, which is more complicated.

Please look at the API description here: <http://docs.servioticy.com/>

- **ServIoTicy actuations** can be tracked by MQTT nodes, as seen on: http://www.servioticy.com/?page_id=273. The node needs to be configured with host *localhost:1883*, username, password and topic *<SOId>/actions*. Afterwards a JSON node needs to be connected on output to handle the correct parsing.
- Matthias Wallnöfer developed nodes for **InTeGreen** (repo: <https://github.com/mwallnoefer/node-red-nodes/commits/integreen>). InTeGreen is the real-time ITS information system of Bolzano/Bozen and is very useful to track traffic and/or ambience situations (delays, parking slots, air pollution ecc...). Usage information here: <http://www.integreen-life.bz.it/>. One node (**InTeGreen In**) is thought to extract general-purpose information, the second one (**InTeGreen ServIoTicy**) should be used

with the ServloTicy Out node for replication purposes.

A service catalog about the offered APIs can be found here:

<http://ipchannels.integreen-life.bz.it/doc/>. Meteorological data for instance is <http://ipchannels.integreen-life.bz.it/MeteoFrontEnd/>, parking slots can be found under <http://ipchannels.integreen-life.bz.it/parkingFrontEnd/>. Please contact the InTeGreen team for more information.

Also here you may access the API without any special node (plain HTTP node), but it will become more complicated.

Now the technical details:

ServloTicy

- input node (data read), returns the latest update
 - attributes *host*, *port*, *soid* (service object ID), *stream*, *channel*, *auth token* (= password)
 - *soid*, *stream*, *channel* are dynamically assignable (by msg.* attributes on input msg)
 - output is JSON-formatted, for instance:
msg.payload = 27.4
msg.lastUpdate = 1409574000
 - *lastUpdate* specified in UNIX timestamp format in [s]
- output node (data write)
 - attributes *host*, *port*, *soid* (service object ID), *stream*, *channel*, *auth token* (= password)
 - *soid*, *stream*, *channel* are dynamically assignable (by msg.* attributes on input msg)
 - *lastUpdate* can be provided as part of the input msg, otherwise “now” will be assumed (UNIX timestamp format in [s])
 - output is JSON-formatted and the confirmation of the correct data acquisition:
msg.payload =
{“channels”: {“<channel>”: {“current-value”: “<value>”}}, “lastUpdate”: <lastUpdate>}
- a UNIX timestamp to plain date output converter for debugging purposes has been developed. Please append “[./convTimeServ.js” onto the curl command. Example:
curl -i -H "Authorization: M2JhMmRkMDEtZTAwZi00ODM5LThtYTktOGU4NjNjYmJmMjc5N2UzNzYwNWltNTc2ZS00MGVILTgyNTMtNTgzMmJhZjA0Zmly" http://localhost:8080/<SO id>/streams/temp1/lastUpdate | ./convTimeServ.js

InTeGreen

- InTeGreen input node (120-integreen)
 - attributes *host*, *port*, *frontend*, *call* and *call parameters*
 - *frontend*, *call* and *call parameters* are dynamically assignable (by msg.* attributes on input msg)

- *call parameters* can be specified in both URI format (*param1=value1¶m2=value2...*) or as JS map (*{ param1: value1, param2: value2... }*)
 - output is JSON-formatted, for *get-records* for instance:
msg.payload =
`[{"timestamp":1409574000000,"value":27.4}, {"timestamp":1409574600000,"value":27.6},...]`
msg.req =
`/MeteoFrontEnd/rest/get-records?station=83200MS&name=LF&seconds=10000`
- InTeGreen ServIoTicy input node which can directly feed the ServIoTicy output node (121-integreen-servioticy)
 - attributes *host*, *port*, *frontend*, *station*, *datatype*, *seconds*
 - *frontend*, *station*, *datatype* and *seconds* are dynamically assignable (by *msg.** attributes on input *msg*)
 - *seconds* may be typed as number or string, but needs to be a valid positive integer
 - output is JSON-formatted and already in ServIoTicy format, for instance:
msg.payload = 27.4
msg.lastUpdate = 1409574000
 - *lastUpdate* specified in UNIX timestamp format in [s]
 - it is suggested to use the InTeGreen-ServIoTicy service object generator *SOgenerator.js* to create a fitting schema for the chosen frontend. The tool has been written by me using Node.js and is released under the Apache 2 license compatible to Node-RED. Usage:
 - *./SOgenerator.js* - asks for the InTeGreen frontend (eg. *MeteoFrontEnd*) and writes output on console
 - *./SOgenerator.js <frontend>* - writes output on console
 - *./SOgenerator.js <frontend> <output>* - writes output to output file
- a UNIX timestamp to plain date output converter for debugging purposes has been developed. Please append "*| ./convTimeInte.js*" onto the curl command. Example:
curl -i
`"http://ipchannels.integreen-life.bz.it/MeteoFrontEnd/rest/get-records?station=83200MS&name=LF&seconds=10000" | ./convTimeInte.js`

Deployment/duplication of pre-existing flows

- **File system:** a copy of the respective flow file into the target Node-RED directory is possible. The file needs to be called *flows_<target hostname>.json*, else a configuration change is necessary. Node-RED can already be running, in which case it needs to be restarted.
- **HTTP:** It is possible to share JSON flow files (usually located under *<node red dir>/flows_<hostname>.json*) to other installations in order to simplify deployments. This can be done using a simple POST request to the active target Node-RED instance. Eg.:

```
curl -X POST -i -H "Content-type: application/json" -d @/tmp/flows_xps732.json  
http://localhost:1880/flows
```