Component Documentation
OSCAR
Operation for Self-checked Co-operative Assembly with Humans and Robots

Organisation name	Country	Type
FlexSight S.r.l.	Italy	Tech
NiTe S.r.l.	Italy	Tech
Alitrak S.r.l.	Italy	SME

Coordinator contact:
Daniele Evangelista
Email: daniele.evangelista@flexsight.eu
Phone: +39 327 7065471

Coordinating DIH:
Pierluigi Cirillo, Ph. D.
Email: cirillo.gigi@gmail.com
Phone: +39 3382069757

Table of Contents

1 OSCAR Documentation

In the following sections, more details are given about the documentation of the whole OSCAR software framework, with particular emphasis on the setup of the FIWARE-ROS2-Unity environment that is the core structure of the entire project (picture of this structure is given in **Error! Reference source not found.**.

1.1 OSCAR – README

This section is in addition to the original version of D4 delivered for the final review meeting of the project, as requested by project coordinators.

The whole OSCAR software framework has been organized in order to follow the structure requested by the DIH2 programme, in particular, packages called “components” are under development to create re-usable software packages in the form of composable tools that together realize the entire OSCAR ROSE-AP.

One of the key component that project partners are developing is the *OSCAR-docker* container. This package has been arranged as a Docker Container for a better organization.

The *OSCAR-docker* container is currently available at: https://hub.docker.com/r/phm14/oscar

N.b. The access to this container requires registration on the official and public Docker HUB cloud space.

Below we resume the readme of this component, also available at the provided link above.

Introduction

The following container **ROS Services for Fiware-UI-Robot communication** is needed to start up the ROS services for OSCAR which enable communication between Fiware, User Interface (UI) and the Robot. More in general OSCAR comprises the following modules:

```
1. CSV Parser
2. ROS Services for Fiware-UI-Robot
3. ROS Services for Kinect-Fiware
4. HoloLens UI application
```

In particular the CSV parser is needed to load the data of the component on Fiware. Fiware, along ROS, is used to enable communication and collection of data across the different devices. Finally the processed data are displayed to the operator through the UI application on HoloLens

Services

After running the ROS launcher the following services will be available:

```
/component_srv: get data concerning the component entity
/step_srv: get data concerning the step entity
/macrostep_srv: get data concerning the macrostep entity
/usecase_srv: get data concerning the use case entity
/step_status: update the step entity with actions performed by the operator
/robot_pose: get data concerning the robot position and load to Fiware
/switch_srv: issue commands to the robot
```

Requirements

CSV Data

Files containing data must be in csv format, using the **comma** `,` delimiter. As for now csv data is split among the following files:

```
Component.csv
Step.csv
Macrostep.csv
UseCase.csv
```

where the entity `UseCase` contains one or more entities `Macrostep`, which, in turn contain one or more entities `Step`.

- The first row of the csv tables contains the attribute name (**first letter in lowercase**) and in the case of relationship attributes their name convention must follow the rule: `name_of_attribute = 'ref' + name_of_external_entity`. So if, for example, an entity `Step` has an attribute linking to the `Component` entity the name of the attribute must be `refComponent`.
- The second row must contain the type of the attribute, the type of the first column containing the `id` must be of type `None`, while the relationship attributes must be of type `Relationship`

Start Fiware

Only the first time execute the command:

```
docker network create fiware_default
```

then run the containers for mongo and orion:

```
docker run -d --name=mongo-db --network=fiware_default   --expose=27017 mongo:4.2 --bind_ip_all
docker run -d --name fiware-orion -h orion --network=fiware_default   -p 1026:1026  fiware/orion -dbhost mongo-db
```

Load data

To load data on Fiware execute the command:

```
python parser_to_fiware.py
```

the script file must be on the same folder of config.yml file to work

Instructions

To run the container execute the command

```
docker run --ipc=host  -it  phm14/oscar
```

Make sure that in the file `config.yaml` inside the folder `src/oscar_core` the entities address field `entities_address: http://IP_ADDRESS:PORT_NUMBER/v2/entities/` is correct:

```
nano src/oscar_core/config.yml
```

and change accordingly `entities_address: http://IP_ADDRESS:PORT_NUMBER/v2/entities/`

Inside the container run the following:

```
source install/setup.bash
ros2 launch bring_up bring_up_launch.py
```

Testing

Inside the container run the following:

```
source install/setup.bash
ros2 launch parser parser_launch.py
```

To check if services are running launch another session from the same container:

```
docker exec -it <container_id> bash
```

inside the new session source ros:`

```
source install/setup.bash
```

To visualize data launch the command:

```
ros2 run parser ficlient TYPE ID
```

where `TYPE` can be one of the following values `component` , `step` , `macrostep` , `usecase` depending on the entity and `ID` is the same value found on the input csv files.

Note

When launching the command `ros2 launch bring_up bring_up_launch.py` the services managing the robot are executed, to work the robot must be connected to the host.
The command `ros2 launch parser parser_launch.py` allow to launch services which communicate with Fiware, in particular entities can be printed on screen using the ficlient node.

1.2 OSCAR - Basic Documentation

This guide instructs the user during the installation of the whole OSCAR framework. It shows the implementation of the ROS2-Fiware bridge and examples of usage regarding one of the main software packages are given, namely the ROS2 parser script that is responsible for parsing the end-user input data regarding the assembly process, moreover this package initializes all the configuration of the OCB and Fiware network.

1.2.1 Prerequisites:

This documentation assumes that the working environment is a Windows OS System with the following prerequisites:

- WSL2 (Windows Linux Subsystem) is installed and running.
- ROS2 Foxy is available and fully configured under WSL2.
- Docker Desktop is installed and running.

1.2.2 Step for Docker Initialization

- Install Docker Desktop: Install Docker Desktop on Windows
- Run Docker Desktop e leave it in background
- Launch Windows terminal (PowerShell is ok too) and run the following commands to initialize the docker and start Fiware:

```
$> docker network create fiware_default
```

```
$> docker run -d --name=mongo-db --network=fiware_default --expose=27017 mongo:4.2 --bind_ip_all
```

```
$> docker run -d --name fiware-orion -h orion --network=fiware_default -p 1026:1026 fiware/orion -dbhost mongo-db
```

If you get an error message like:

```
$> Error response from daemon: network with name fiware_default already exists
```

Run the following command to "reset" Docker:

```
$> docker system prune
```

the output (which we simply need to confirm by typing "y") will be something like this:

```
$> WARNING! This will remove:
$>    - all stopped containers
$>    - all networks not used by at least one container
$>    - all dangling images
$>    - all dangling build cache
$> Are you sure you want to continue? [y/N]
```

1.2.3 Configure ROS Foxy on WSL2

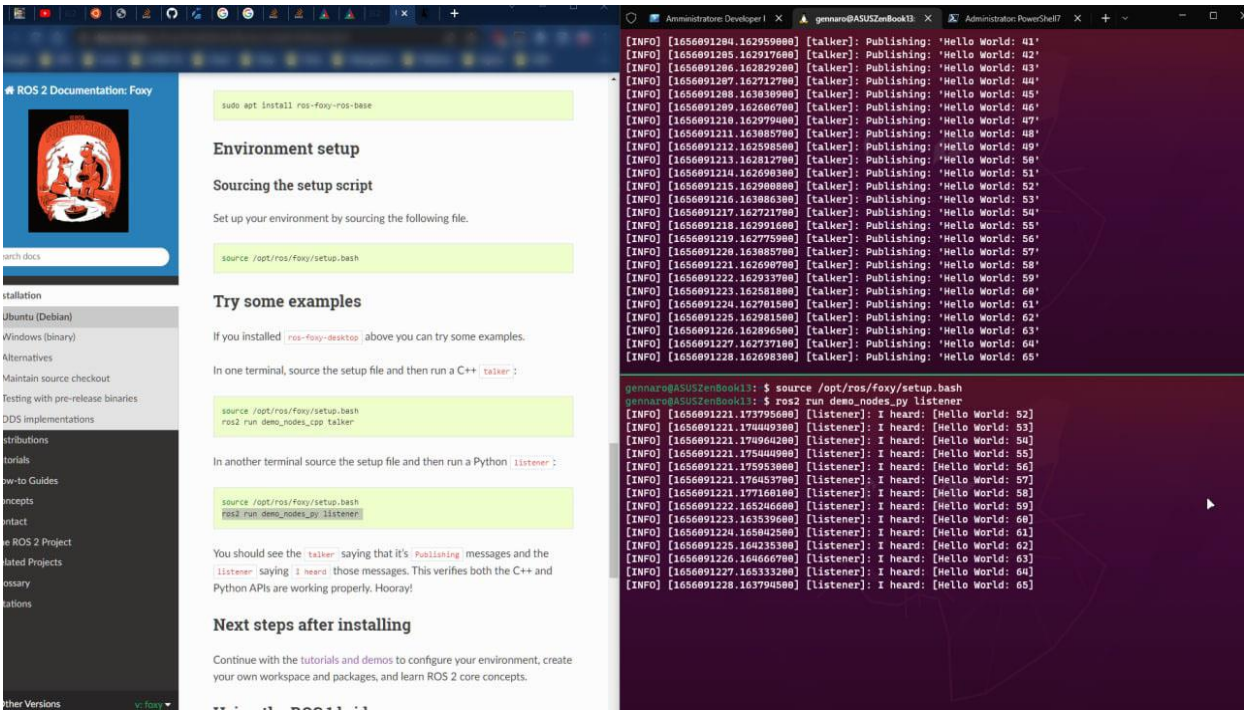
This part is a tutorial on how to set up a working ROS development environment within Windows (complete with GUI support) utilizing the new Windows Subsystem for Linux 2 (WSL2).

- Open a windows terminal as administrator
- Check that the WSL version is 2 by default with this command: `wsl --set-default-version 2`
- Check out distro available online: `wsl --list –online`
- Install **Ubuntu-20.04**: `wsl --install -d Ubuntu-20.04`
- Here you must configure Ubuntu, put username and password, and possibly restart the PC.

- To check wsl distros and theri version: `wsl --list --verbose`. The output should be something like that:

```
  NAME                   STATE           VERSION
* Ubuntu-20.04           Running         2
  Ubuntu-22.04           Running         2
  docker-desktop         Running         2
  docker-desktop-data    Running         2
```

- Launch **Ubuntu-20.04** under WSL2
- Install ROS2 Foxy: [ROS2 Ubuntu]
- Source ROS2 Foxy: `source /opt/ros/foxy/setup.bash`
- Follow the instructions from the documentation untill you can run:
  `ros2 run demo_nodes_cpp talker`
  and
  `ros2 run demo_nodes_py listener`
  as shown below:



1.2.4 Setup the workspace and deploy the OSCAR code repository

- Create a folder and call it Oscar; this is our *workspace*
- Install `python3-colcon` to build packages following this guide: [Using colcon to build packages]
- Create a `src` folder inside `Oscar`
- Clone (requires FlexSight approval) the OSCAR code repository inside `src` with:

  ```
  $> git clone git@bitbucket.org:flexsight/oscar_core.git
  ```

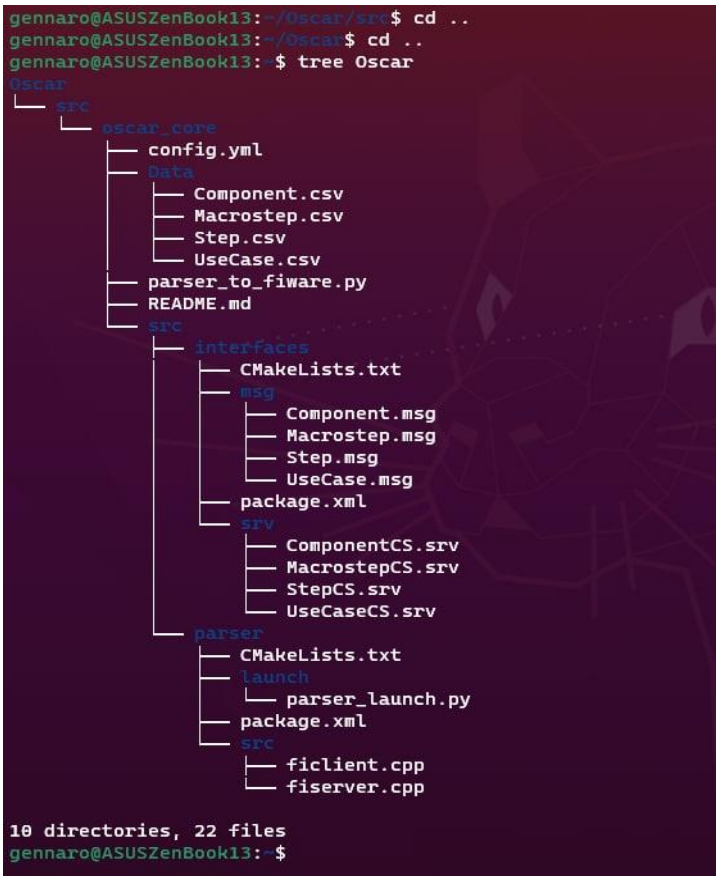
- Your project structure should be something like:

  ```
  Oscar/
  ├── src
  │   ├── oscar_core
  │   │   ├── README.md
  ```

- Inside `oscar_core` use this command to switch to the correct branch:

  ```
  $> git checkout hotfix/windows
  ```

- Your project structure should look like this:



1.2.5 Upload company input data into FIWARE

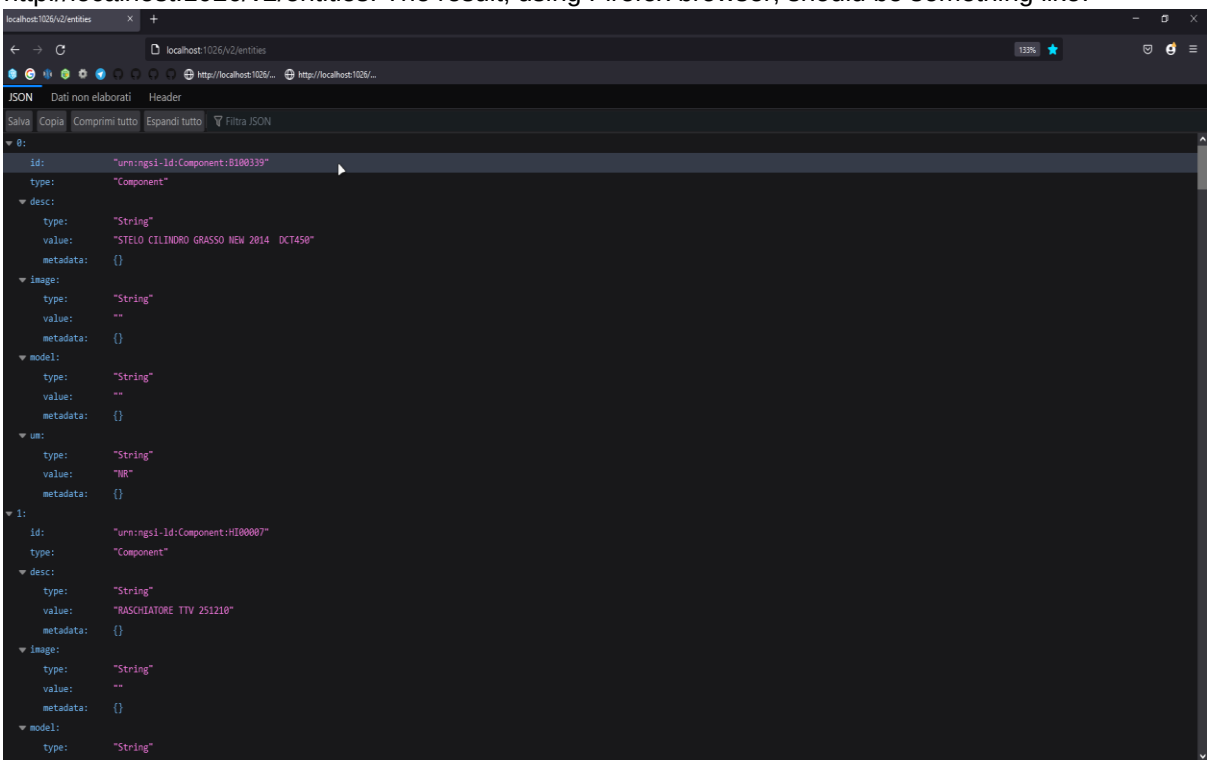
The script `parser_to_fiware.py` is required to upload your data into FIWARE.

- Inside `oscar_core`, in the main project ROS2 workspace, run the python script with the following command:
  `$> python3 parser_to_fiware.py`

  The output should be something like that:

```
$> Loading process data on Fiware
$> ********************************
$> Component was uploaded successfully
$> Step was uploaded successfully
$> Macrostep was uploaded successfully
$> UseCase was uploaded successfully
$> ********************************
$> Loading empty robot pose on Fiware
$> ********************************
$> Robot pose was uploaded successfully
$> ********************************
```

- Check that the data has been uploaded to FIWARE by opening the following browser address: http://localhost:1026/v2/entities. The result, using Firefox browser, should be something like:



1.2.6 Compile the ROS2 Packages

- Move in the main workspace `Oscar`
- Source ROS2 Foxy main installation: `source /opt/ros/foxy/setup.bash`
- Execute `colcon build` to compile

NOTE: the first time the ROS packages are compiled, the operation can take up to a few minutes. If everything has been configured correctly and the compilation was successful, the following output is obtained:

```
$> Starting >>> interfaces
$> Finished <<< interfaces [0.93s]
$> Starting >>> parser
$> Finished <<< parser [0.59s]
$> Summary: 2 packages finished [1.81s]
```

If the compilation fails, you can run the following command to show a more detailed output log:

```
$> colcon build --event-handlers console_direct+
```

1.2.7 Quick Trial Test

In the same terminal where the colcon build command was launched, it is possible to initialize the ROS2 server node:

1. Source the installation: `source install/setup.bash`
2. Launch parser server: `ros2 launch parser parser_launch.py`

If everything go fine, you will get a similar output:

```
$> [INFO] [launch]: All log files can be found below /home/gennaro/.ros/log/2022-06-27-18-11-08-969783-ASUSZenBook13-9180
$> [INFO] [launch]: Default logging verbosity is set to INFO
$> [INFO] [fiserver-1]: process started with pid [9182]
```

Now it's time to initialize a client node that queries the server node; therefore:

1. Open a new Ubuntu terminal
2. Source ROS2 Foxy: `source /opt/ros/foxy/setup.bash`
3. Move in the workspace folder (Oscar)
4. Source the installation: `source install/setup.bash`
5. To visualize data launch the command: `ros2 run parser ficlient TYPE ID`

Where:

- `TYPE` can be one of the following values: `component, step, macrostep or usecase`, depending on the entity.
- `ID` is the same value found on the input .csv files (You can try `0` or `1`).

For example, the output of the command:

```
$> ros2 run parser ficlient macrostep 0
```

according to the data structure, will be the following:

```
Steps:
   ID: 0
   Component:
      ID: B100339
      Desc: STELO CILINDRO GRASSO NEW 2014  DCT450
      UM: NR
      Image:
      Model:

   Robot: 0
   Quantity: 1
   Image:
```

```
   Model:

   ID: 1
   Component:
      ID: HI00007
      Desc: RASCHIATORE TTV 251210
      UM: NR
      Image:
      Model:

   Robot: 0
   Quantity: 1
   Image:
   Model:

   ID: 2
   Component:
      ID: B100293ZN
      Desc: BLOCCHETTO ANTIROTAZIONE CILINDRO GRASSO DCT450
      UM: NR
      Image:
      Model:

   Robot: 0
   Quantity: 1
   Image:
   Model:

   ID: 3
   Component:
      ID: U5927M10X12
      Desc: GRANO CAVA ESAGONALE PUNTA CONICA UNI 5927M10X12
      UM: NR
      Image:
      Model:

   Robot: 0
   Quantity: 3
   Image:
   Model:

   ID: 4
   Component:
      ID: U7435D25
      Desc: ANELLO SEEGER PER ALBERI  25  UNI 7435 D25
      UM: NR
      Image:
      Model:

   Robot: 0
   Quantity: 1
   Image:
   Model:

Image:
Model:
```

1.3 OSCAR – Step-by-Step Tutorial

In this section we provide a brief tutorial on how to run and start the Oscar main applications, namely the software interface between ROS2 and Fiware and the bridge that connects ROS2 with the mixed-reality Unity environment. For a better understanding of the whole software structure, please look at **Error! Reference source not found.**.

1.3.1 Prerequisites

Once you have configured the Windows environment (Docker Desktop) and WSL2 with Ubuntu 20.04 and ROS2 Foxy as described in Section 0, follow these steps to start the whole application

1.3.2 Initialize ROS2 server node

In a new command window in the Ubuntu environment:

1. Move in the main workspace `Oscar`
2. Source ROS2 Foxy main installation: `source /opt/ros/foxy/setup.bash`
3. Source the installation: `source install/setup.bash`
4. Launch parser server: `ros2 launch parser parser_launch.py`

You should obtain the following output:

```
$> [INFO] [launch]: Default logging verbosity is set to INFO
$> [INFO] [fiserver-1]: process started with pid [9182]
```

Note: if an error message appears, you may be asked to compile the packages from scratch. If this is the case, please run the following steps:

1. Move in the main ROS2 workspace `Oscar`
2. Source ROS2 Foxy main installation: `source /opt/ros/foxy/setup.bash`
3. Run `colcon build` to compile

Note: the first time the ROS packages are compiled, this operation can take up to a few minutes. If everything has been configured correctly and the compilation was successful, the following output is obtained:

```
$> Starting >>> interfaces
$> Finished <<< interfaces [0.93s]
$> Starting >>> parser
$> Finished <<< parser [0.59s]
$> Summary: 2 packages finished [1.81s]
```

1.3.3 Launch the ROS2-Unity bridge

Open a new command window in the Ubuntu environment:

1. Move in the main workspace `Oscar`
2. Source ROS2 Foxy main installation: `source /opt/ros/foxy/setup.bash`
3. Source the installation: `source install/setup.bash`
4. Determine your IP using the following command: `hostname -I`
5. Run the following command, replacing `<your IP address>` with your ROS machine's IP or hostname:
   `$> ros2 run ros_tcp_endpoint default_server_endpoint --ros-args -p ROS_IP:=<your IP address>`

Note: If everything went well you will get an output similar to this:

```
$> [INFO] [1664188490.726480800] [UnityEndpoint]: Starting server on <your IP address>:10000
```

Note: If this command doesn't work, try replacing:

```
$> default_server_endpoint --ros-args -p ROS_IP:=<your IP address>
```

H2020 Innovation Action – This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824964