D3.11

Interoperability Report Y2

Grant agreement no.	780785
Project acronym	OFERA (micro-ROS)
Project full title	Open Framework for Embedded Robot Applications
Deliverable number	D3.11
Deliverable name	Interoperability Report
Date	December 2019
Dissemination level	Public
Workpackage and task	3.3
Author	Borja Outerelo Gamarra (eProsima)
Contributors	
Keywords	Micro-ROS, Report, Robotics, ROS, microcontrollers, middleware, Interoperability, ROS2, FIWARE, SOSS
Abstract	This document provides a report regarding the interoperability of micro-ROS with existing ROS2 and with FIWARE.

Contents

1 Acronyms

Acronym	Explanation
CDR	Common Data Representation
DDS	Data Distribution Service
DDS-XRCE	DDS For Extremely Resource Constrained Environments
GA	Grant Agreement
LTS	Long Term Support
OFERA	Open Framework for Embedded Robotic Applications
ROS	The Robot Operating System
RTPS	Real Time Publish Subscribe
SOSS	System Of Systems Synthesizer
WP	Work package

2 Executive summary

This report, **D3.11 Interoperability Report - Y2**, provides an overview of the status of the interoperability of micro-ROS, on the second year of the project.

OFERA consortium focused on two micro-ROS interoperability points during the reporting period:

- Interoperability with ROS 2 and ROS systems.
- Interoperability with FIWARE via SOSS.

In the previous year, there was a third interoperability point, Hardware interoperability. This reporting period, Hardware interoperability, has been halted as the Consortium member working on it has left the project. Hardware interoperability cornerstone was the use of HRIM, however, eProsima, which will take over the rest of Acutronics tasks, at the moment is not going to continue with HRIM.

Regarding ROS 2 and ROS system interoperability, OFERA consortium has been adopting more ROS 2 concepts and updated existing developments to be aligned and interoperable with the new ROS 2 releases, including the first LTS release of ROS 2. Maintaining the initial architecture approach allows keeping micro-ROS and ROS 2 interoperable at different levels.

In micro-ROS architecture definition, the interoperability with current ROS 2 middleware protocol, DDS/RTPS, was taken into account. The middleware protocol that consortium chose for micro-ROS, DDS-XRCE, provides this DDS interoperability.

OMG creates a DDS-XRCE protocol with this interoperability as one of its primary objectives. DDS-XRCE grants interoperability between its clients and existing DDS global data spaces. DDS-XRCE achieves this interoperability thanks to the use of a DDS-XRCE Agent which acts as a bridge between DDS-XRCE networks and a DDS global data space.

For all the previous, using DDS-XRCE and granting DDS interoperability allows the interoperability between micro-ROS and ROS 2 systems, in the current status of ROS 2 implementations. The interoperability with ROS 1, this year can come from two different sources: 1) using existing ROS 2-ROS 1 bridge

or 2) using SOSS with ROS 2 system handler and ROS system handler. SOSS is the new standard way to interoperate with ROS 2, it has been developed by EPROS and OSRF during this year.

WP3 work and more precisely task 3.3: ROS bridging & interoperability achieved this ROS2 and ROS interoperability.

micro-ROS approach to the interoperability with FIWARE has changed during this year. As mentioned before, SOSS is to become the new ROS Integration Services standard, connecting, apart of ROS 1 and ROS 2, any other protocol with ROS 2. In this year, the consortium has been focused on adopting this new mechanism to communicate with FIWARE. Analogous to the approach for ROS 1 interoperability, FIWARE-ROS 2 communication is done using SOSS core along with two system handlers, one for ROS 2 and other for FIWARE Context broker. This way and following a similar architecture compared to last year, micro-ROS is able to interoperate with FIWARE thanks to its ROS 2 interoperability. SOSS does not allow only interoperability through ROS 2. Native DDS System Handler has been developed so any application using DDS-XRCE can communicate with any other application connected to SSOS using any of the available system handlers.

WP3 involving task 3.4: Fiware interoperability achieved this FIWARE interoperability.

3 Introduction

3.1 Purpose of document

The purpose of this document is to provide an overview of the interoperability of the Horizon 2020 Innovation Action shortly named OFERA - Grant Agreement Number 780785 o the European Commission. It covers the second period of the project, from 1st January 2019 to 31st December 2019 (twelve months, from M13 to M24). This interoperability report targets two different systems and its micro-ROS interoperations: ROS 2 (ROS), and FIWARE. Hardware interoperation, which depends on HRIM has been dropped from this report as Acutronic left the project and the partner taking over most of their tasks, eProsima, at the moment has not any development plans on the existing HRIM work.

This report follows the following structure:

- Part 1 is the executive summary with a quick review of the advances done.
- Part 2 is the present introduction.
- Part 3 is the report itself. It ...
  - Explains ROS 2 interoperability
  - Explains FIWARE interoperability
- Part 5 contains conclusions.

This document is a revised version of last year periodic report. OFERA has one last release scheduled for the project, as shown below:

- Interoperability Report Y3 - 30.06.2020

4 Interoperability report

In this section, the report defines and explains the interoperability with ROS 2 and FIWARE, regarding their status at the end of this first year of OFERA project.

4.1 ROS 2 Interoperability

4.1.1 Introduction

micro-ROS interoperability with ROS2 was one of the technical objectives of micro-ROS as mentioned in the OFERA GA. Also, it could be seen as the must-have interoperability as it builds one of the foundations of micro-ROS, bringing ROS 2 to embedded devices.

ROS 2 interoperability and portability has been a critical piece on the design and development of the current micro-ROS version. The current version of micro-ROS pursues to resemble as much as possible the usage of ROS 2. For that reason, micro-ROS reuses the same concepts of ROS 2, and even It reuses a significant amount of the tooling surrounding ROS 2.

This reuse of existing concepts and tools allows micro-ROS to achieve a high level of integration with current ROS 2 systems. In this revision, apart of continue using and evolving the micro-ROS architecture approach, micro-ROS reuses ROS 2 build system and types definitions. These two concrete cases ease, to a great extent, the interoperability between micro-ROS and ROS 2:

a. The reuse of build system allows having the same process for generating interfaces and their associated and required libraries. In this direction, a new micro-ROS-build package has been created to locate all micro-ROS build configuration in only one package. With this new package user can build, micro-ROS-Agent, micro-ROS-demos and cross-compile for the different platforms within a ROS 2 environment. This way working with a micro-ROS workspace does not differ from regular work with ROS 2 workspaces.
b. The reuse of existing types definitions ensures that the generation of code using them as the base, uses the same source as in ROS 2, avoiding types mismatching issues. During this year, two new versions of ROS 2 has been released, Dashing Diademata and Eloquent Elusor. Dashing Diademata is the first LTS release of ROS 2. One of the changes in ROS 2, this year release, is a change in the type support, in which OSRF has decided to use DDS standard definition language for types, IDL. This will ease future support from the micro-ROS part, as Micro XRCE-DDS uses that same standard IDL.

Apart from those high-level integrations and message definitions, micro-ROS proposes to use standard middleware protocols, natively interoperable with the one used in ROS 2, DDS.

The protocol chosen for micro-ROS is the OMG DDS-XRCE as it is a perfect match with DDS, both come from the same standardisation body as well as DDS-XRCE was designed and created with DDS interoperation as a foundational requirement. micro-ROS can be interoperable with any DDS vendor. Even, interoperation between DDS-XRCE client and agent is possible between different vendors as DDS-XRCE standardise that communication between Agent and Client.

Interoperation of DDS different vendors is out of the scope of this project. Still, OMG standardisation was designed with interoperability in mind providing multiple levels of interoperation. From the DDS

interface, through the interoperable wire protocol used, RTPS, to the data representation, CDR. DDS vendors and users continuously test this DDS vendor interoperability in different ways; two of them are worthy of mentioning here:

1. OMG official GitHub repository [1] with interoperability tests. DDS vendors manually run these tests.
2. ROS 2 official CI site with automated tests [2], using different rmw implementations OSRF runs those tests on a CI tool, Jenkins. Sample results of an execution: Jenkins test run

The use of such standardised and DDS interoperable protocols, as this report explains later eases the interoperability between ROS 2 and micro-ROS at the middleware layer. Despite changes in DDS-XRCE standard done this year, these standard changes keep DDS interoperability untouched.

4.1.2 micro-ROS current status

As introduced above, OFERA achieved micro-ROS and ROS 2 interoperability from the early stages of the project. This interoperability was eased thanks to the use of interoperable middleware protocols, DDS and DDS-XRCE on ROS 2 and micro-ROS side respectively.

As already mentioned, the current ROS 2 implementation uses OMG DDS as the underlying middleware protocol. Using DDS middleware allows any DDS implementation to be interoperable, almost out of the box, with ROS 2. Consortium partner eProsima provides one of the DDS implementations, Fast RTPS, and even it has been chosen to be the ROS 2 default middleware implementation.

The eProsima situation, as the ROS 2 default middleware providers and its participation as an OMG member on the standardisation of these protocols, provides micro-ROS with a good advantage position. eProsima, apart from DDS implementation, also develops and maintains the current micro-ROS middleware implementation, Micro XRCE-DDS which implements DDS-XRCE. Being implementer of both ends in the interoperability, Fast RTPS and Micro XRCE-DDS, ensures that both products will keep interoperable from early development stages to the latest standard modifications. Users and eProsima proved this interoperability in different ways:

a. eProsima uses CI with integration tests [3]. These tests use Micro XRCE-DDS Agents and Clients, along with Fast RTPS testing communication between all the pieces.
b. Third parties: Renesas and Robotis are actively using micro XRCE-DDS. They mainly use Micro XRCE-DDS to communicate with ROS 2 (DDS).
    - Renesas enables their platform GR-ROSE to be a ROS 2 participant using Micro XRCE-DDS [4].
    - Robotis uses Micro XRCE-DDS as a bridge to communicate their XEL Network with ROS 2 [5].

During this reporting period, Micro XRCE-DDS has been ported and adapted to new Fast RPS releases, 1.8 and 1.9 which are the core for Dashing Diademata and Eloquent Elusor ROS 2 releases. Adapting Micro XRCE-DDS Agent to the use this new Fast RTPS versions ensures middleware interoperability with ROS 2 different versions.

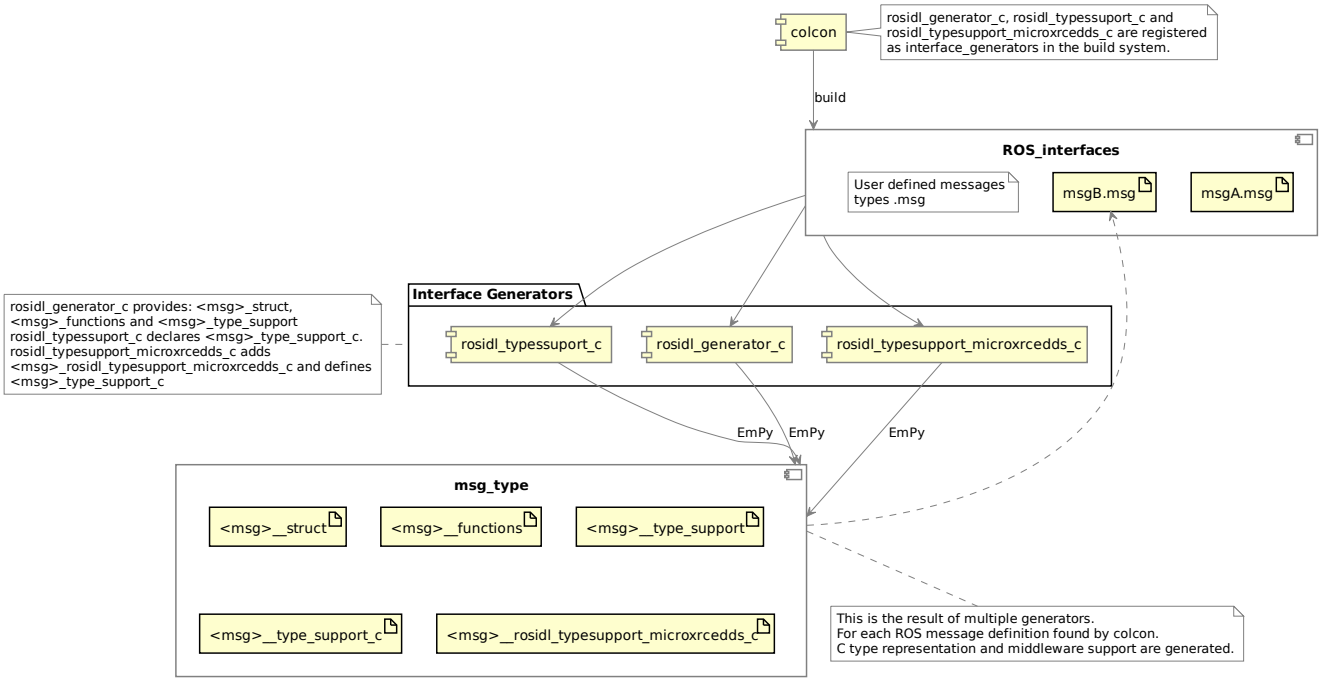
Even though, this middleware interoperability is the base to a ROS2 - micro-ROS interoperability is not the only interoperability needed for achieving full integration. One of the critical aspects of ROS 2 is the standardisation of their interfaces, simplifying, the interoperability between applications. ROS 2 interface plays a prominent role in the interoperability, as an addition to the last year publish-subscribe interface, services have been supported as part of the work of this year. Supporting services included a standardisation change in DDS-XRCE in which an RPC mechanism has been introduced. This new request-reply model has also been implemented in the upper layers of micro-ROS.

During this year and as mentioned before, ROS 2 types have suffered some changes in the latest release. The changes didn't affect the work developed last year apart of the need to port build time mechanism. This year micro-ROS adapted to support the new IDL format supported by ROS 2 build time tools. micro-ROS currently partly support the types and same type definition as ROS 2 does. There is support for all the basic types and some of the vectorised ones. micro-ROS does not support all the complex type vectors as could be arrays of strings and similar "complex types". This new type support was added as part of a new release supporting LTS version of ROS 2 Dashing Diademata. Next steps in this matter will be to provide complete type support for micro-ROS. This type support allows any type from ROS 2 to be ported to micro-ROS providing interoperability of the interfaces using it.

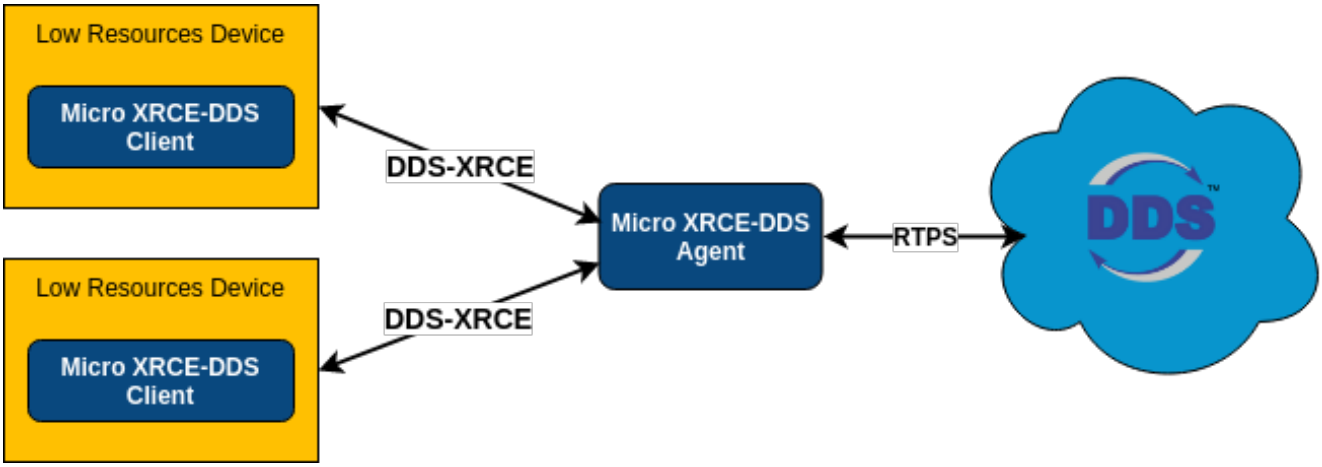
OFERA have provided demonstrations of ROS 2 messages from and to micro-ROS systems in the current micro-ROS version. The consortium keep updating these demonstrations in micro-ROS-demos These demonstrations use the mechanism mentioned above where automatically generation of type support is achieved on both ends, ROS 2 and micro-ROS, using the same source type definition file. These demonstrations show the users how to include new types and how straightforward it would be to integrate any types in both ends of the communication, in this case, micro-ROS and ROS2.

4.1.3 Architecture

ROS 2 interoperation is directly related to type support and middleware implementations. In the case of micro-ROS, consortium members have developed a type support mechanism similar to the one used in ROS 2. In micro-ROS, build system generates type support libraries in C language for each interface found in the workspace. The source of the generation is precisely the same as in ROS 2 that allows the reuse of types definition is easing the port of types from one end to the other end of the interoperable system. The following figure shows this interface generation process.



As stated above, another key point to allow micro-ROS - ROS 2 interoperability was the usage of interoperable middleware. On the ROS 2 end, the middleware used is based on DDS. On the other side, micro-ROS has been architecture and developed using a new OMG standard 100% interoperable with DDS. This new standard is DDS-XRCE, allowing to communicate embedded devices with DDS networks natively. DDS-XRCE provides the interoperability between micro-ROS devices with existing ROS 2 applications at middleware level.



This reuse of type definition together with the usage of interoperable middleware provides micro-ROS with native interoperation with ROS 2. From this point a wide range of interoperation for micro-ROS is opened, for example in the case of ROS 1, using the existing bridge from ROS 2 to ROS 1 will ensure that micro-ROS could interoperate with ROS 1 system hopping over ROS 2. Also, one of the most significant changes regarding interoperability comes in this same area; this year, SOSS has been developed. It is explained further later, but it could be summarised as the integration tool for any kind of communication mechanism with ROS 2, and as stated before, it provides access to those communications to micro-ROS

hopping over ROS 2.

4.2 FIWARE Interoperability (FIWARE System Handler and SOSS)

4.2.1 Introduction

This year work in FIWARE interoperability has taken a new architectural approach. This year System of Systems Synthesizer (SOSS), has been developed and released. SOSS provides a set of tools and configuration utilities to communicate different communication middlewares. SOSS is based in a plugin system (In this case called System Handlers) and a common data representation, which is OMG Extensible and Dynamic Topic Types for DDS (DDS-XTYPES). OSRF plans to adopt SOSS as the default ROS 2 integration mechanism, providing a well-defined way to connect ROS 2 with any other system. Also, a ROS-Industrial FTP outcome: ROS2 Integration Service, initial release uses SOSS (System Of Systems Synthesizer) as core communications enabler.

During this year, previous year interoperability has been changed to use this new tool, SOSS, adding FIWARE System Handler (Plugin enabling FIWARE communication in SOSS). For this second year, micro-ROS interoperability with FIWARE is done using SOSS utilities and getting advantage of probed micro-ROS - ROS 2 interoperability. FIWARE - micro-ROS interoperability is done via ROS 2 (DDS) the same way as the previous ROS interoperability, hopping over ROS 2.

4.2.2 micro-ROS current status

As mentioned before, micro-ROS during this second year has adopted the new SOSS package. Using SOSS and thanks to the micro-ROS - ROS 2 interoperability micro-ROS is able to interoperate, not only with FIWARE but with any system capable of being integrated with SOSS. Currently SOSS supports: ROS 1, ROS 2, DDS, Websocket and FIWARE, with more integrations under development.

micro-ROS - FIWARE integration was achieved using SOSS and two of its System Handlers, ROS 2 and FIWARE. In this time and thanks to SOSS, the integration was easier compared to last year as each of the system handlers has the specific details of their protocols, NGSIv2 in the case of FIWARE and DDS topics in case of ROS 2. This abstraction of the other end of the communication is one of the strong points to develop SOSS, adding integrations will be easier.

4.2.3 Architecture

SOSS is based in a plugin system. In this system there is a first step in which the user configures the behaviour of the communications using a YAML file. In this YAML file, the communication channels between System Handlers are defined. And a second phase where OMG DDS-XTYPES takes a key role in being the standard data representation that each system Handlers interact with.

The SOSS architecture can be summarised with the following diagram.

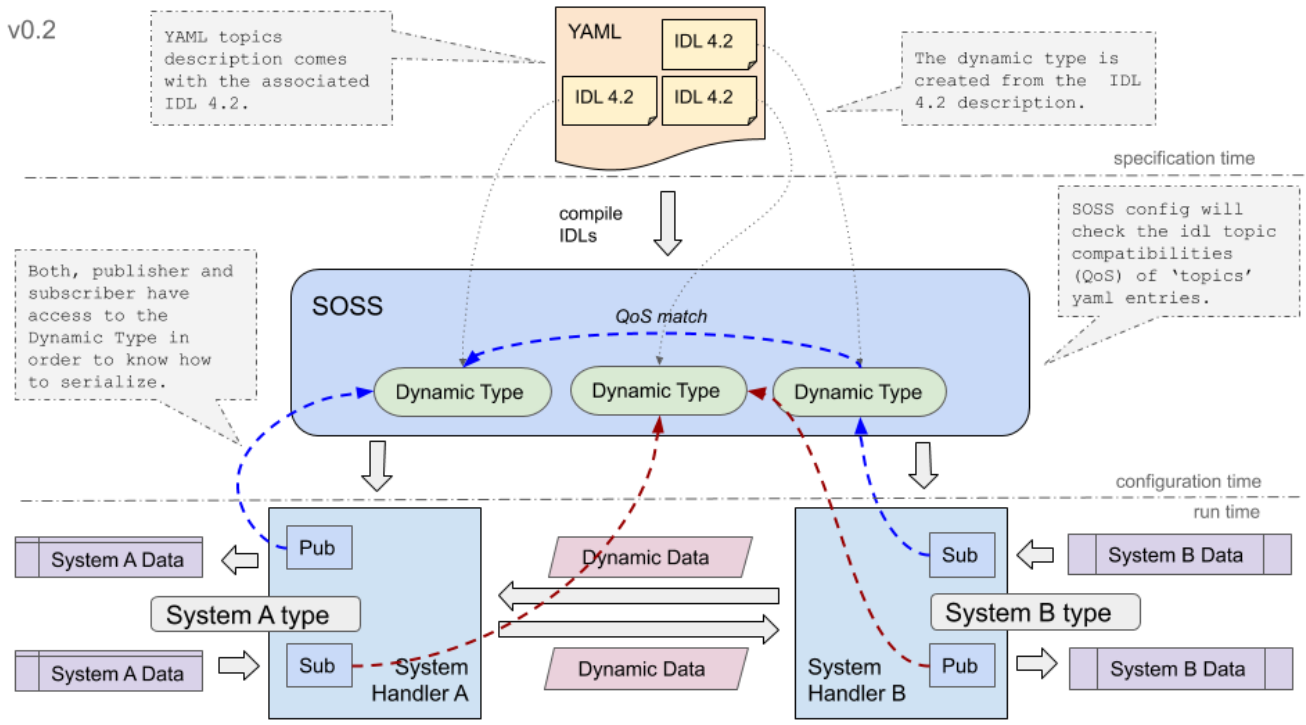


Figure 1: SOSS full architecture

SOSS has been developed with the idea to be integrated not only with new ROS 2 installations, but with existing ROS 2 ones. For that, new type support has been developed in which SOSS generates dynamic types based on existing ROS 2 types. In the following diagram, this mechanism and the files generated are detailed.

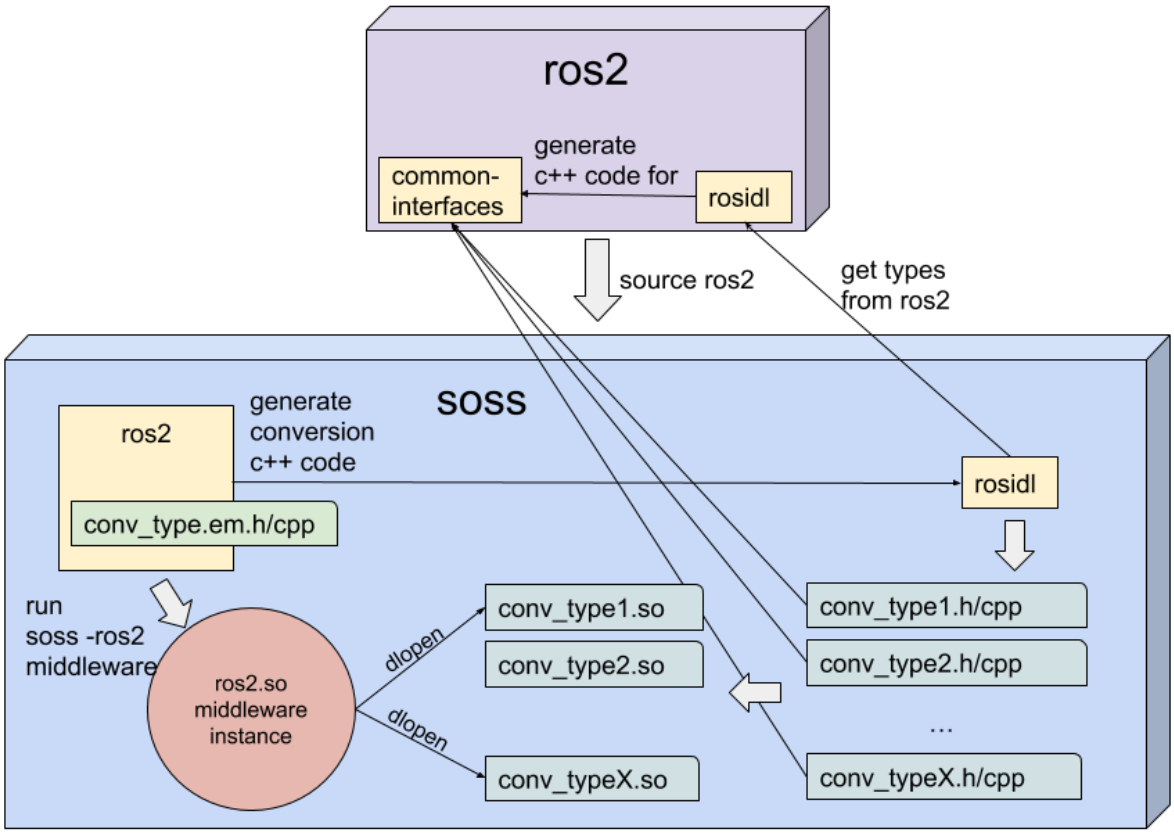


Figure 2: SOSS type support

micro-ROS integration with FIWARE gets the advantage of this new system. The micro-ROS - FIWARE Context broker integration is done using SOSS along with ROS 2 System Handler and FIWARE System Handler. micro-ROS has “native” integration with ROS 2. In this case, micro-ROS communicates with ROS 2 without the need of any System Handler or SOSS intervention. Them thanks to the use of SOSS and a FIWARE System Handler, ROS 2 can publish and subscribe to FIWARE Context broker data. As micro-ROS subscriptions and publications from outside are seen as regular ROS 2 subscriptions and publications, FIWARE System Handler can make use of them out of the box

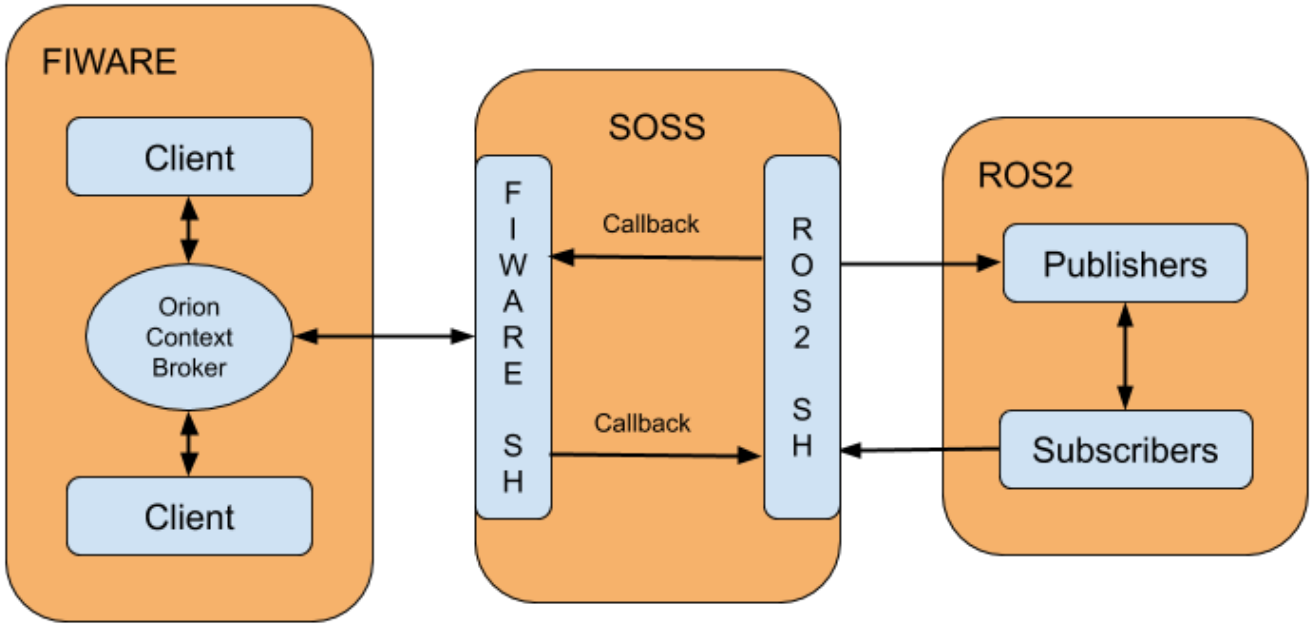


Figure 3: SOSS ROS 2 - FIWARE

4.3 Conclusion

During the reporting period, ROS 2 interoperability has been slightly increased, from the types and interfaces perspective. FIWARE interoperability has been rearchitected around a new software release.

ROS2 and FIWARE interoperability were achieved relatively easy thanks to the use of standard protocol (DDS and DDS-XRCE). One of the most time-consuming efforts was aligning version with ROS 2 releases, but with the already support of ROS 2 LTS version achieved this year, the work for future releases will be easier.

ROS 2 integration needs further development to support all the data types. However developing services was a big step towards full interoperability, development of interfaces needs to be done, more precisely, parameters and events are two ROS 2 concepts that will ensure better interoperability between micro-ROS and current ROS 2 systems.

From the FIWARE point of view, interoperability has been achieved with the same extension as the one between ROS 2 and FIWARE. Future native micro-ROS interoperability will be achieved developing a micro-ROS system handler which will allow micro-ROS to communicate with any other System integrated into SOSS.

The following steps in this sense will be to develop further ROS 2 concepts and develop a micro-ROS System Handler for SOSS integrations.

References

[1] O. (Object Management Group), 'Interoperability examples for the data-distribution service (dds) standard'. 2018 [Online]. Available: https://github.com/omg-dds

[2] O. (Open Source Robotics Foundation), 'Official ros 2 communication tests'. 2018 [Online]. Available: https://github.com/ros2/system_tests/tree/master/test_communication

[3] eProsima, 'Micro xrce-dds integration tests'. 2018 [Online]. Available: https://github.com/eProsima/Micro-XRCE-DDS-Integration-Tests

[4] godzilla-max, 'A dds-xrce implementation for rx65n'. 2018 [Online]. Available: https://github.com/godzilla-max/rose_sketch

[5] Robotis, 'Robotis xel network integration with micro xrce-dds'. 2018 [Online]. Available: https://xelnetwork.readthedocs.io/en/latest/