SROS 2

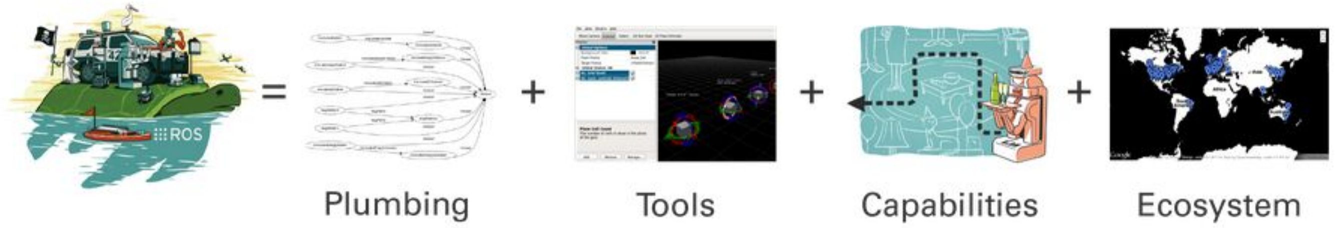
Mikael Arguedas
IROS 2018, Madrid

open
robotics

SROS2

- What is ROS 2
- Interfacing DDS-Security to the ROS 2 stack
- Use the sros2 command line interface
- Run some basic examples

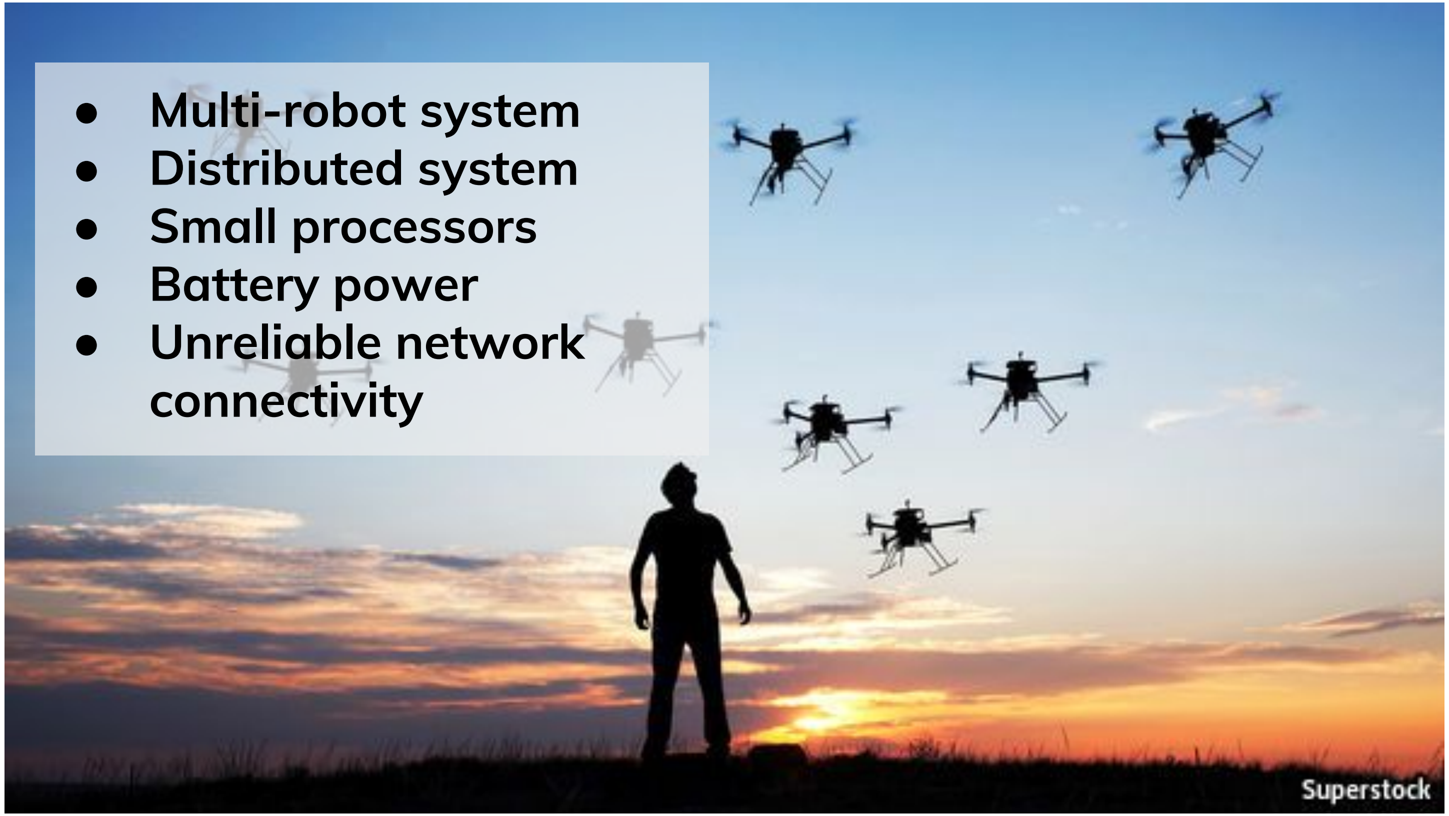
ROS as we know it



Characteristics of systems initially targeted by ROS in 2007

- Research applications
- High-volume sensors
- Complex kinematics
- Lots of computation power
- Ideal network connectivity

Characteristics of small robotic systems today

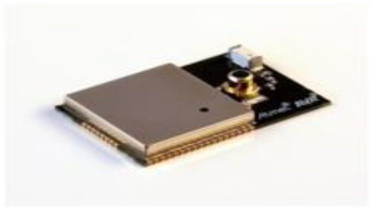


- Multi-robot system
- Distributed system
- Small processors
- Battery power
- Unreliable network connectivity

Goals of ROS 2

Support multi-robot systems involving unreliable networks

Remove the gap between prototyping and final products


"Bare-metal" micro controller

Support for real-time control

Cross-platform support

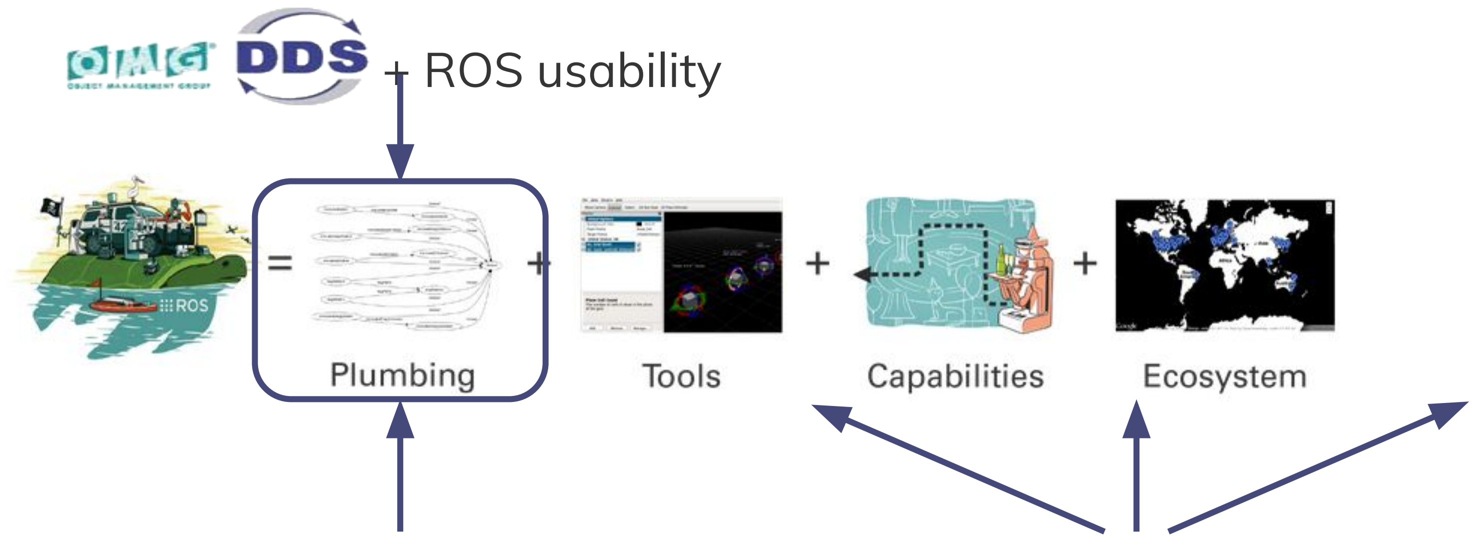
http://design.ros2.org/articles/why_ros2.html

open robotics

ROS 2

ROS
=
Plumbing
+
Tools
+
Capabilities
+
Ecosystem

ROS 2



less time spent here

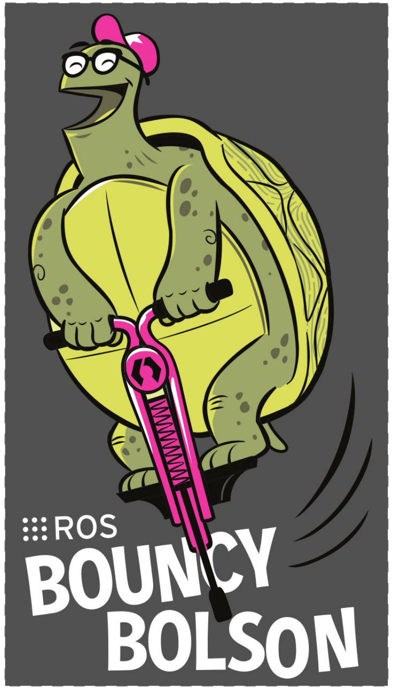
means

more time to spend here

ROS 2 Releases

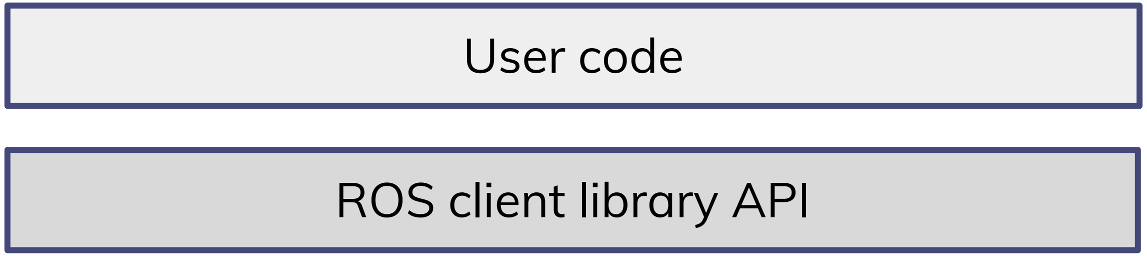
December 2017

June 2018

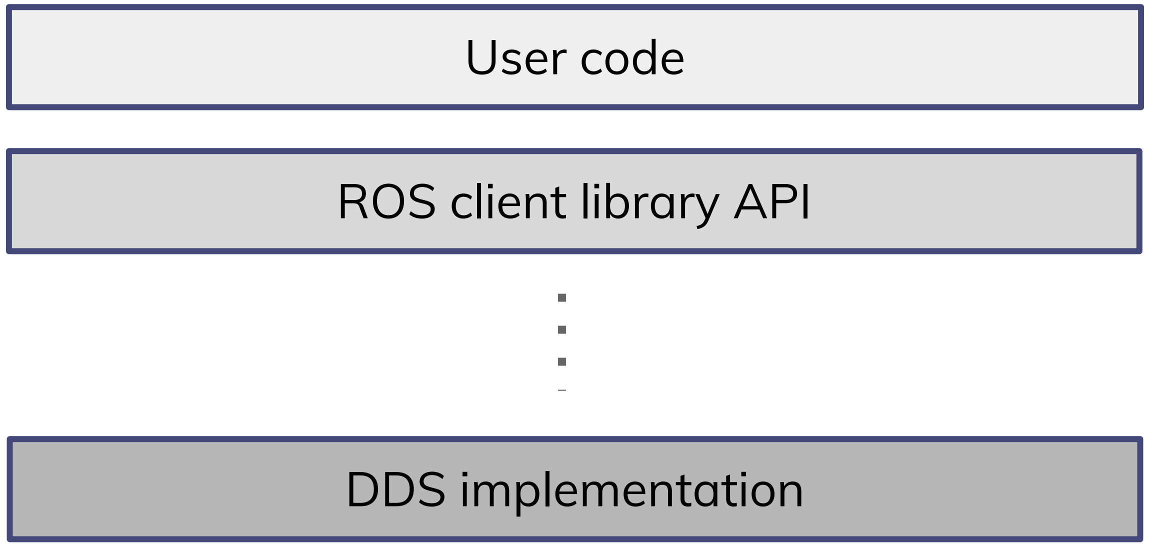


open robotics

Architectural overview

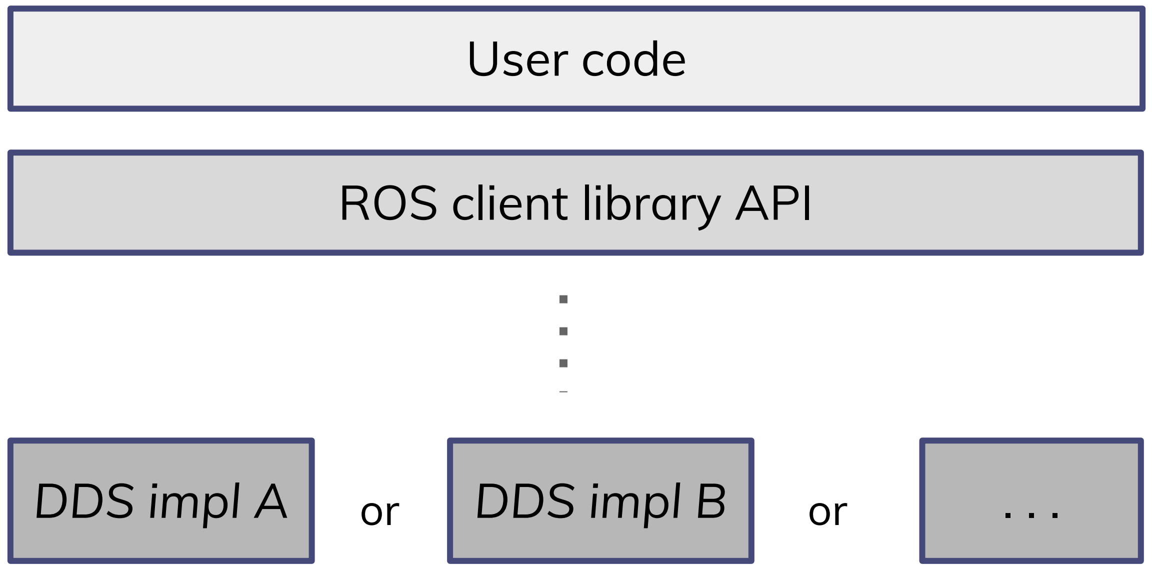


Architectural overview



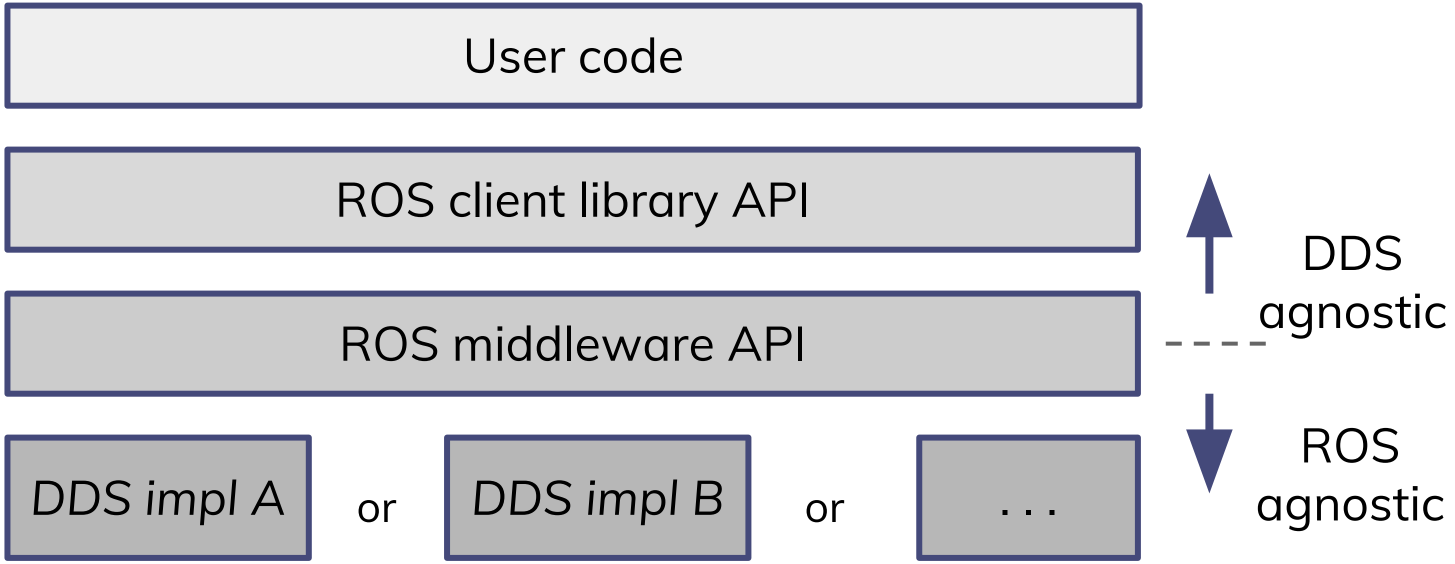
= discovery + serialization + transport

Architectural overview



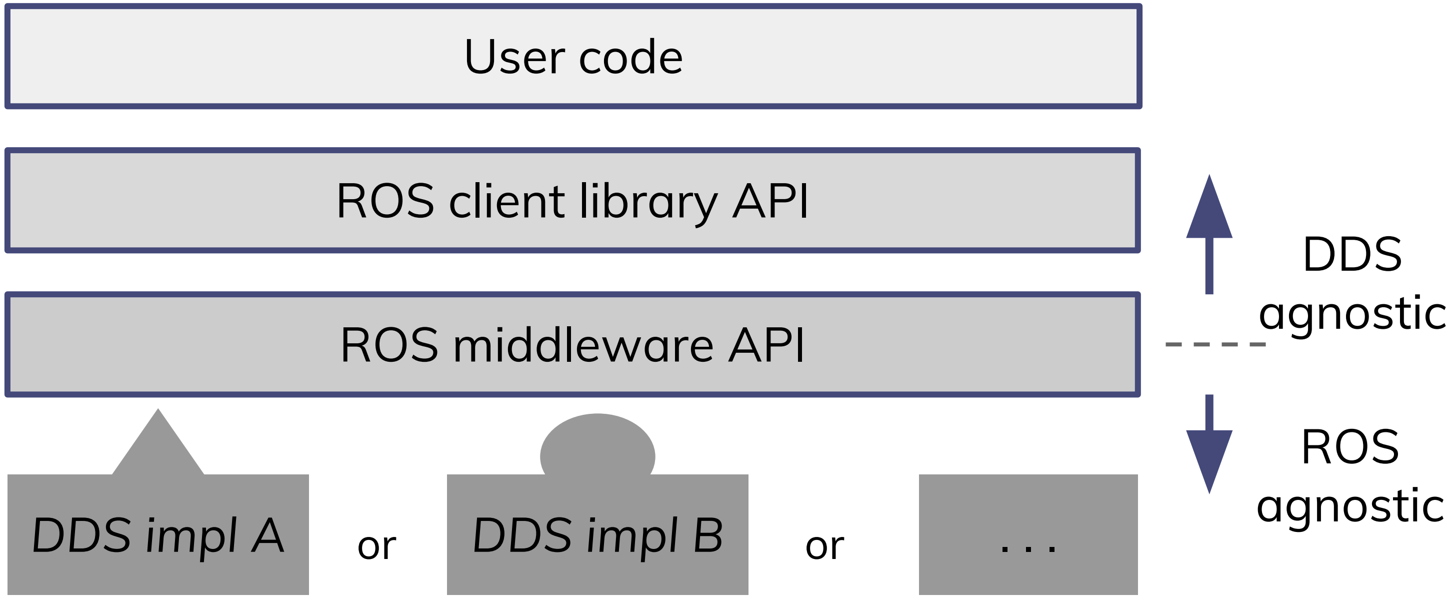
open robotics

Architectural overview

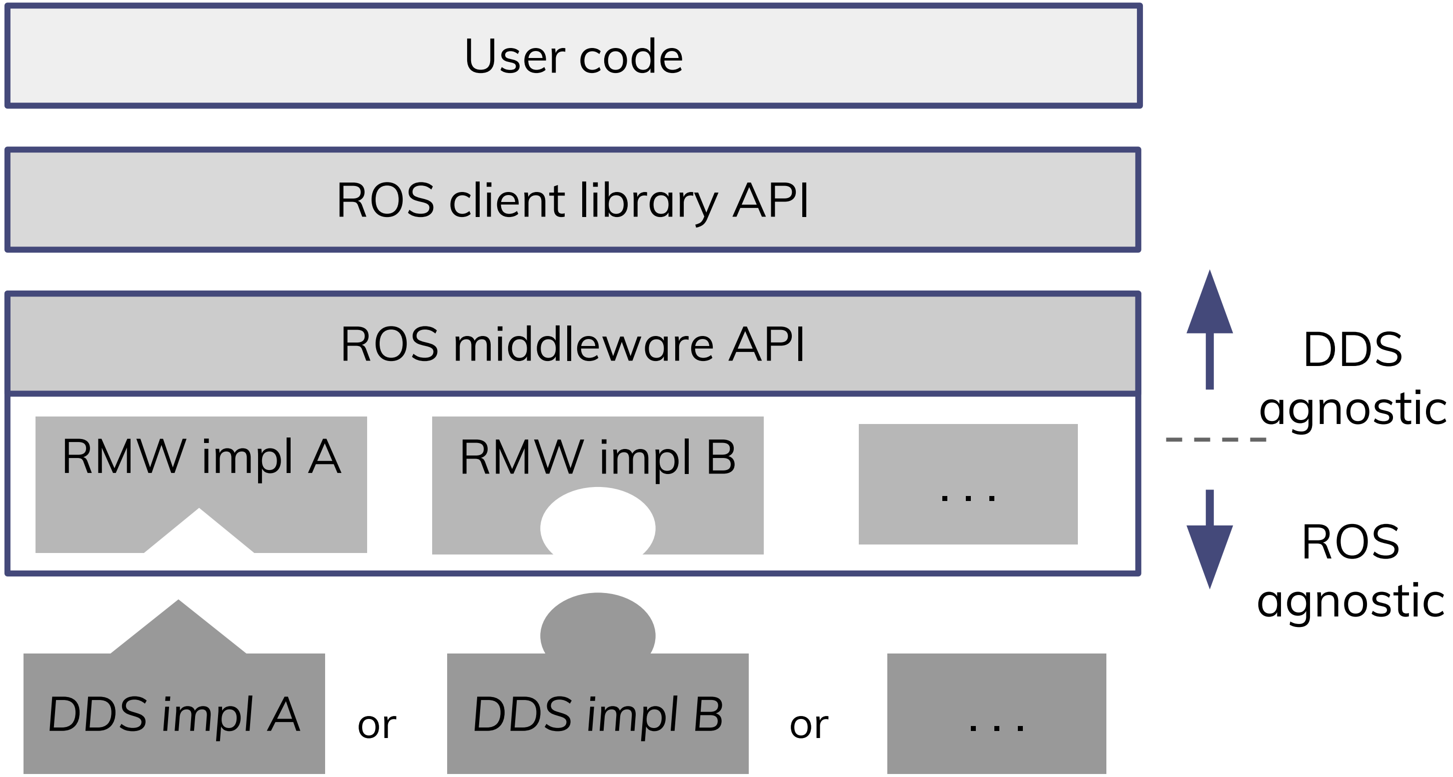


open robotics

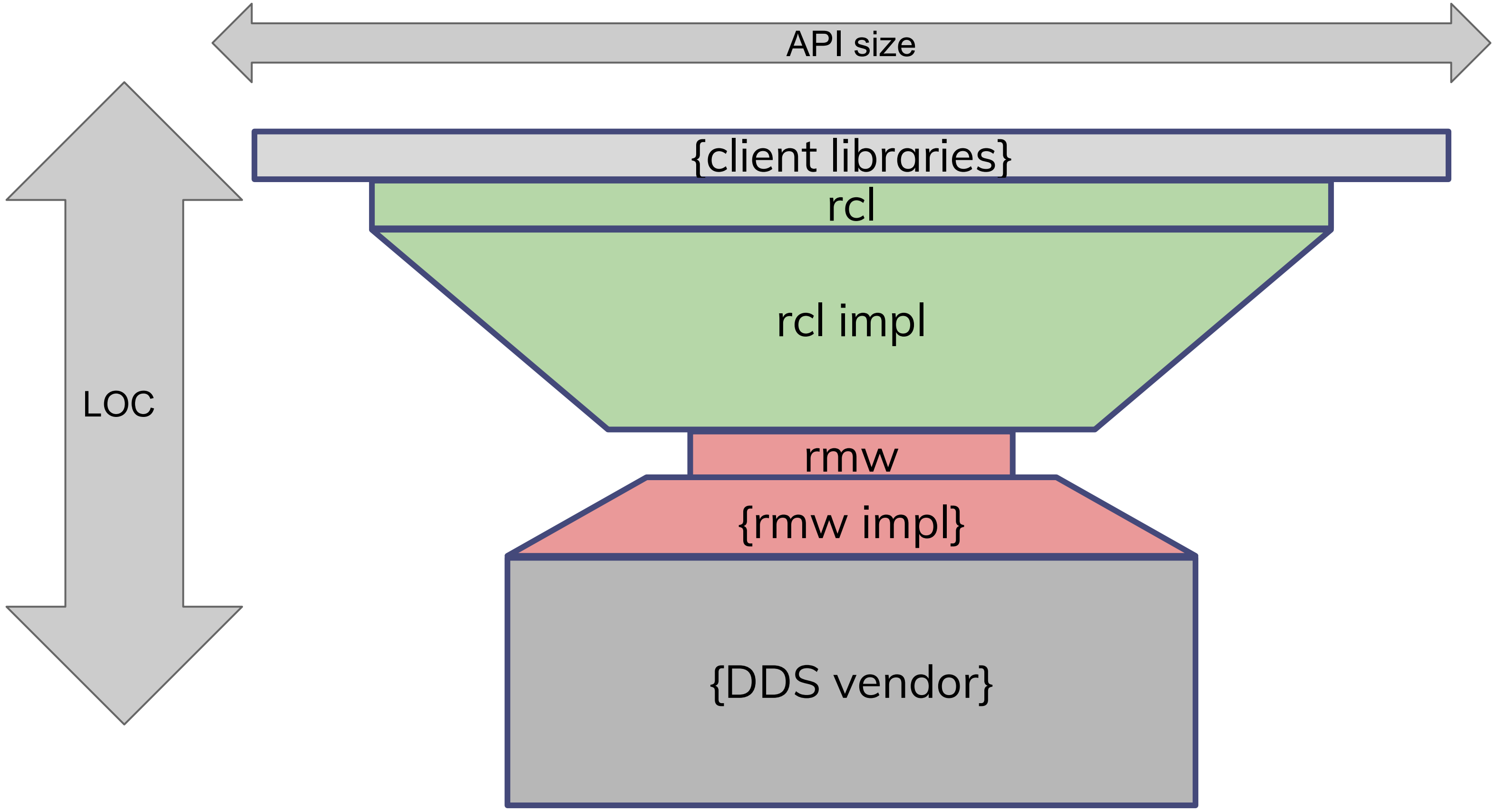
Architectural overview



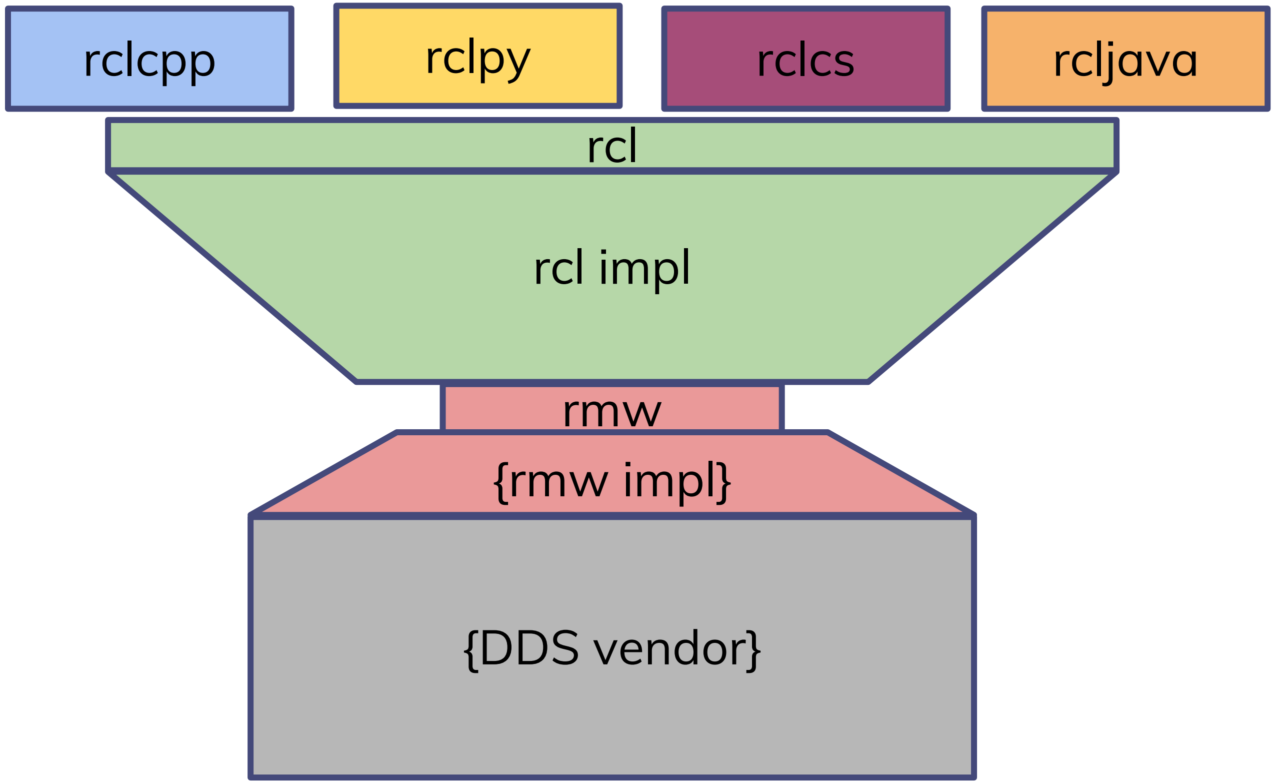
Architectural overview



“Hour Glass” Pattern

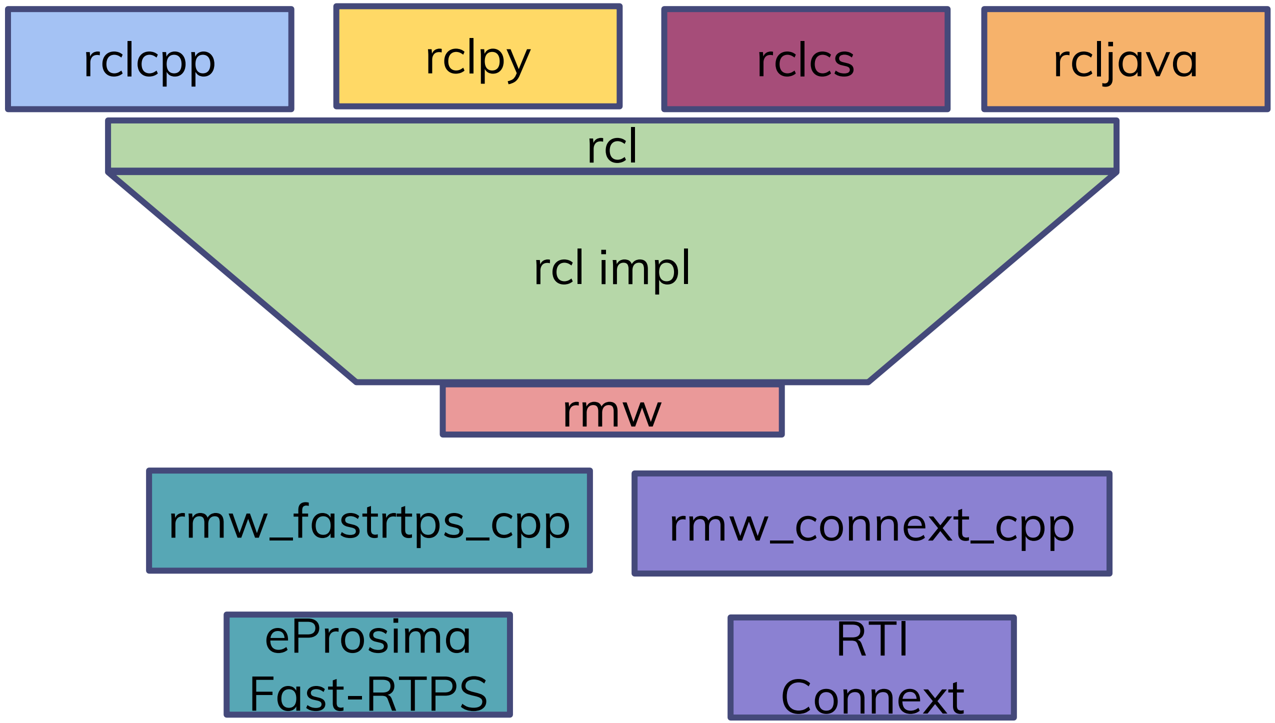


“Hour Glass” Pattern



open robotics

“Hour Glass” Pattern



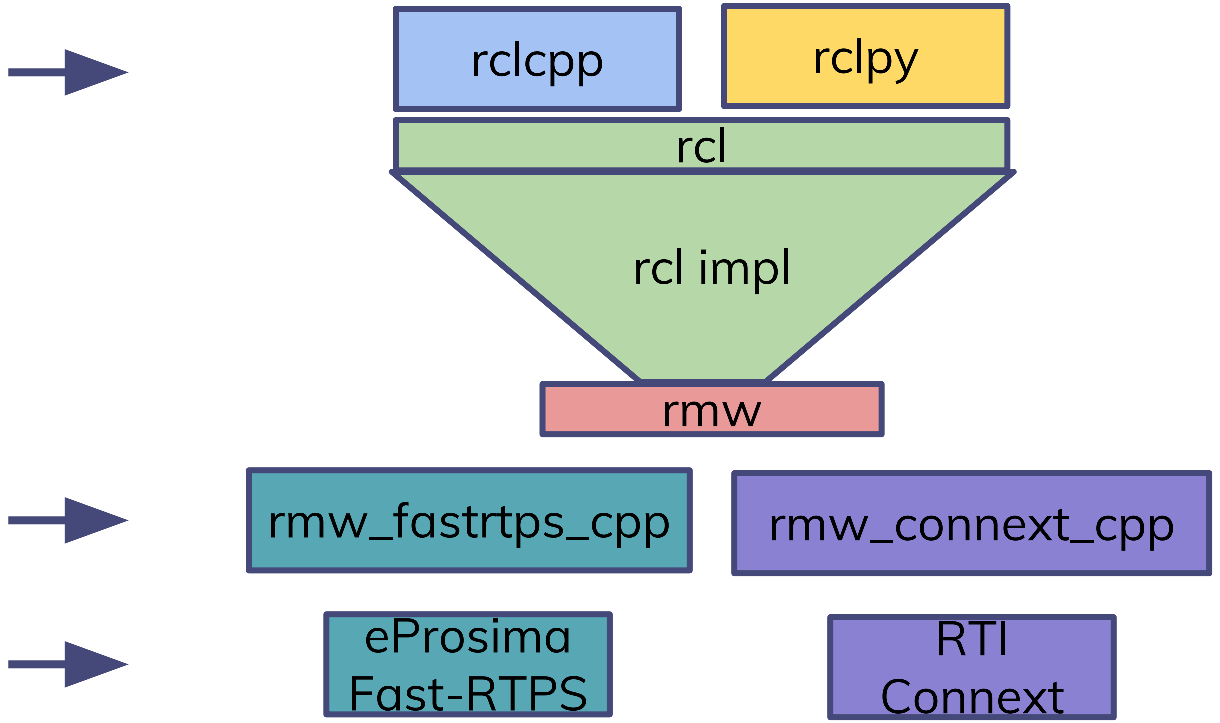
open robotics

Where does SROS 2 live?

User code should **not** change

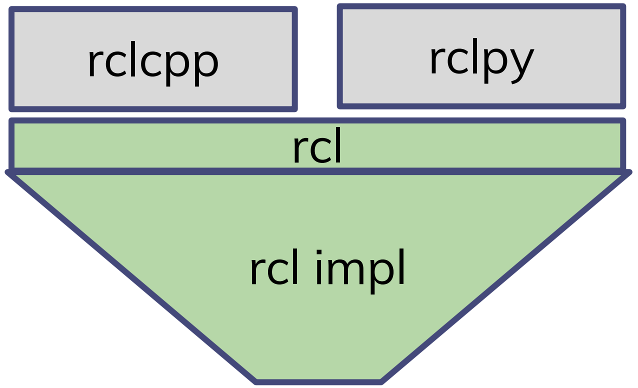
Plugin instantiation

DDS-Security implementation



Where does SROS 2 live?

Environment variables checking



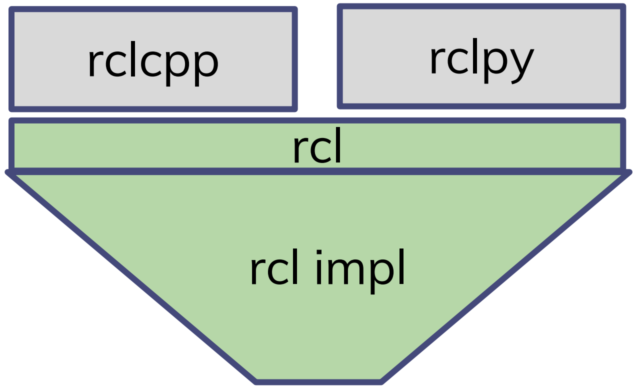
ROS_SECURITY_ENABLE → Should we look for security artifacts ?

ROS_SECURITY_STRATEGY → Should we prevent unauthenticated nodes from being created ?

ROS_SECURITY_ROOT_DIRECTORY → Where to look for artifacts

Where does SROS 2 live?

Environment variables checking

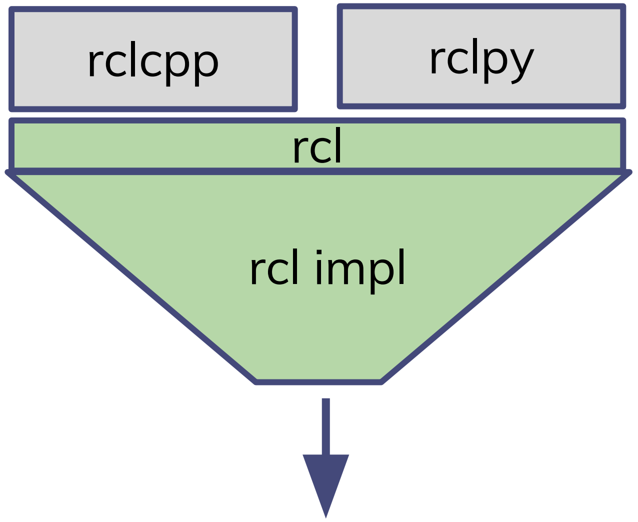


ROS_SECURITY_ENABLE →	true/false
ROS_SECURITY_STRATEGY →	Permissive/Enforce
ROS_SECURITY_ROOT_DIRECTORY →	<path/to/keystore>

open robotics

Where does SROS 2 live?

Environment variables checking
+
Keystore node structure checking

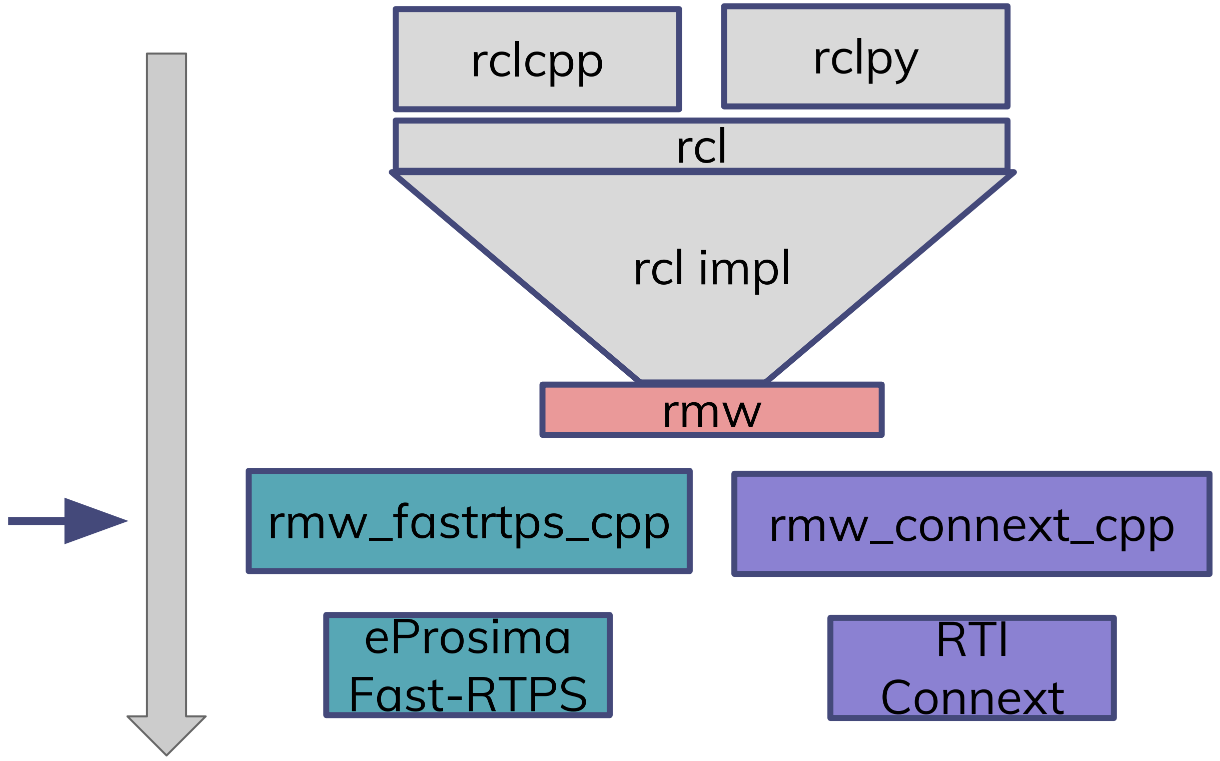


Extract node security directory path

open robotics

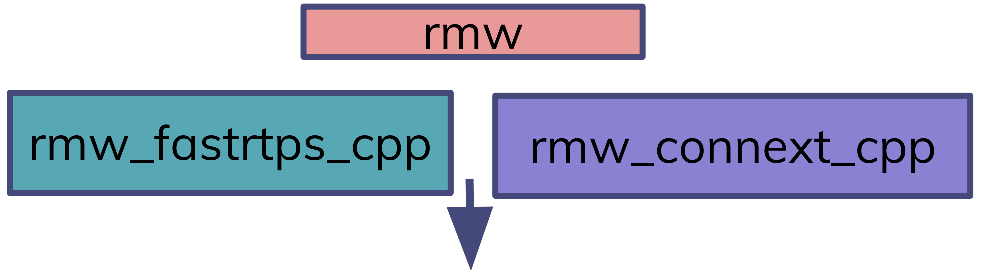
Where does SROS 2 live?

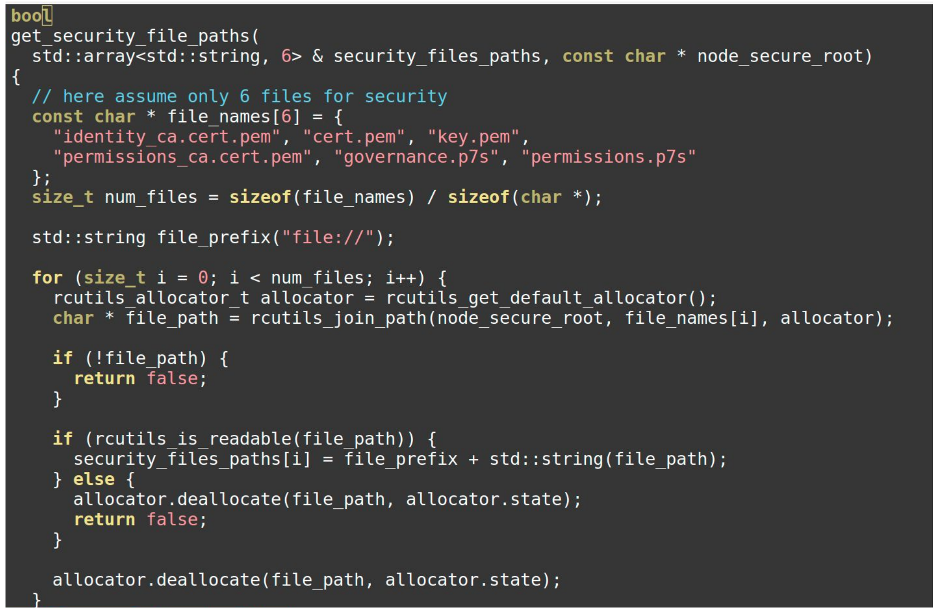
Retrieve security artifacts
+
Instantiate plugins accordingly



Where does SROS 2 live?

Retrieve security artifacts

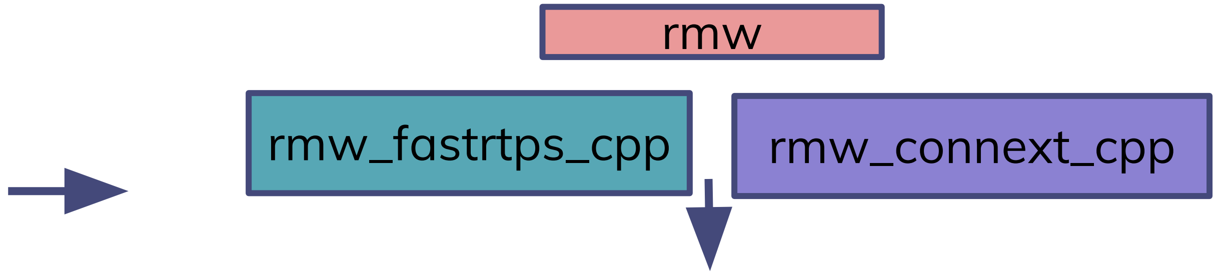


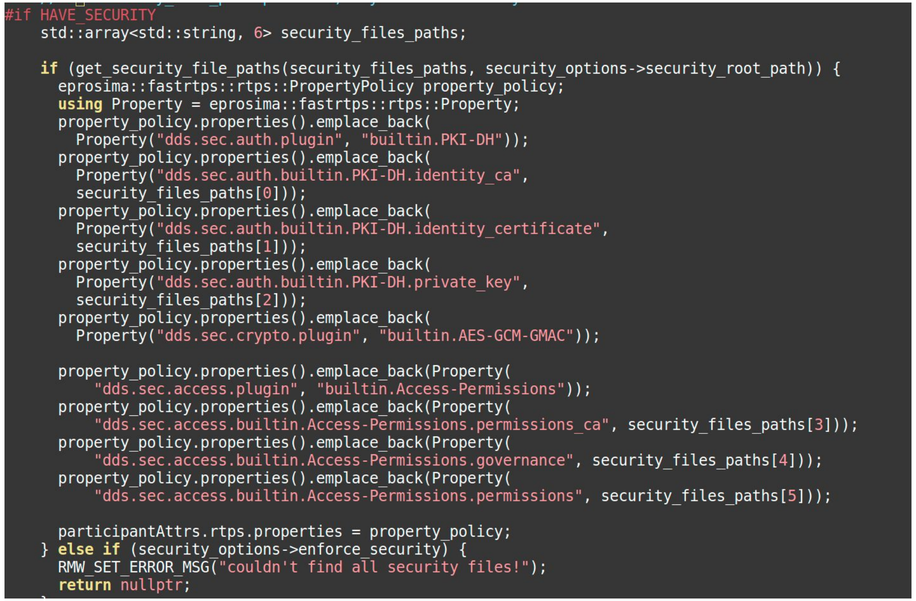


```
bool
get_security_file_paths(
  std::array<std::string, 6> & security_files_paths, const char * node_secure_root)
{
  // here assume only 6 files for security
  const char * file_names[6] = {
    "identity_ca.cert.pem", "cert.pem", "key.pem",
    "permissions_ca.cert.pem", "governance.p7s", "permissions.p7s"
  };
  size_t num_files = sizeof(file_names) / sizeof(char *);

  std::string file_prefix("file://");

  for (size_t i = 0; i < num_files; i++) {
    rcutils_allocator_t allocator = rcutils_get_default_allocator();
    char * file_path = rcutils_join_path(node_secure_root, file_names[i], allocator);

    if (!file_path) {
      return false;
    }

    if (rcutils_is_readable(file_path)) {
      security_files_paths[i] = file_prefix + std::string(file_path);
    } else {
      allocator.deallocate(file_path, allocator.state);
      return false;
    }

    allocator.deallocate(file_path, allocator.state);
  }
```

Where does SROS 2 live?

Instantiate security plugins



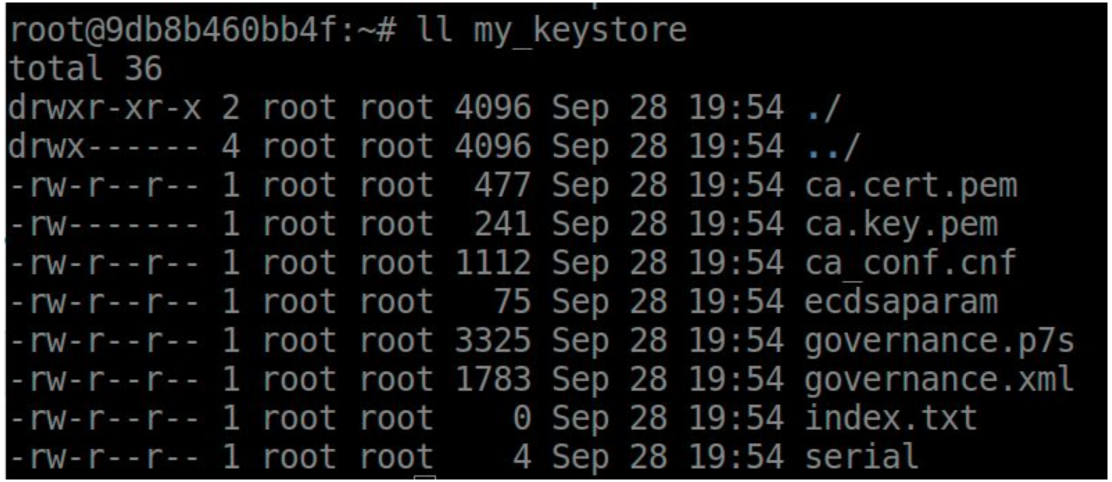


```
#if HAVE_SECURITY
    std::array<std::string, 6> security_files_paths;

    if (get_security_file_paths(security_files_paths, security_options->security_root_path)) {
      eprosima::fastrtps::rtps::PropertyPolicy property_policy;
      using Property = eprosima::fastrtps::rtps::Property;
      property_policy.properties().emplace_back(
        Property("dds.sec.auth.plugin", "builtin.PKI-DH"));
      property_policy.properties().emplace_back(
        Property("dds.sec.auth.builtin.PKI-DH.identity_ca",
        security_files_paths[0]));
      property_policy.properties().emplace_back(
        Property("dds.sec.auth.builtin.PKI-DH.identity_certificate",
        security_files_paths[1]));
      property_policy.properties().emplace_back(
        Property("dds.sec.auth.builtin.PKI-DH.private_key",
        security_files_paths[2]));
      property_policy.properties().emplace_back(
        Property("dds.sec.crypto.plugin", "builtin.AES-GCM-GMAC"));

      property_policy.properties().emplace_back(Property(
          "dds.sec.access.plugin", "builtin.Access-Permissions"));
      property_policy.properties().emplace_back(Property(
          "dds.sec.access.builtin.Access-Permissions.permissions_ca", security_files_paths[3]));
      property_policy.properties().emplace_back(Property(
          "dds.sec.access.builtin.Access-Permissions.governance", security_files_paths[4]));
      property_policy.properties().emplace_back(Property(
          "dds.sec.access.builtin.Access-Permissions.permissions", security_files_paths[5]));

      participantAttrs.rtps.properties = property_policy;
    } else if (security_options->enforce_security) {
      RMW_SET_ERROR_MSG("couldn't find all security files!");
      return nullptr;
```

How to generate SROS 2 artifacts?

- Setting up your environment:
  - $ source /opt/ros/bouncy/setup.bash

- Create a keystore:
  - $ ros2 security create_keystore my_keystore

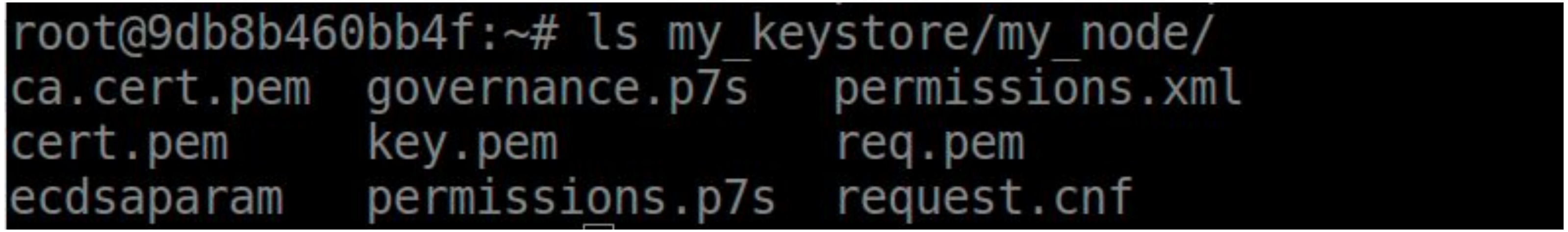


```
root@9db8b460bb4f:~# ll my_keystore
total 36
drwxr-xr-x 2 root root 4096 Sep 28 19:54 ./
drwx------ 4 root root 4096 Sep 28 19:54 ../
-rw-r--r-- 1 root root  477 Sep 28 19:54 ca.cert.pem
-rw------- 1 root root  241 Sep 28 19:54 ca.key.pem
-rw-r--r-- 1 root root 1112 Sep 28 19:54 ca_conf.cnf
-rw-r--r-- 1 root root   75 Sep 28 19:54 ecdsaparam
-rw-r--r-- 1 root root 3325 Sep 28 19:54 governance.p7s
-rw-r--r-- 1 root root 1783 Sep 28 19:54 governance.xml
-rw-r--r-- 1 root root    0 Sep 28 19:54 index.txt
-rw-r--r-- 1 root root    4 Sep 28 19:54 serial
```

open robotics

How to generate SROS 2 artifacts?

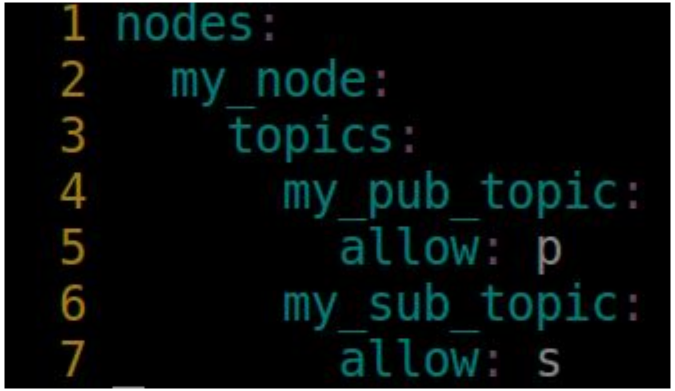
- Create key and wildcard permissions for a node:
  - $ ros2 security create_key my_keystore my_node



```
root@9db8b460bb4f:~# ls my_keystore/my_node/
ca.cert.pem  governance.p7s   permissions.xml
cert.pem     key.pem          req.pem
ecdsaparam   permissions.p7s  request.cnf
```

How to generate SROS 2 artifacts?

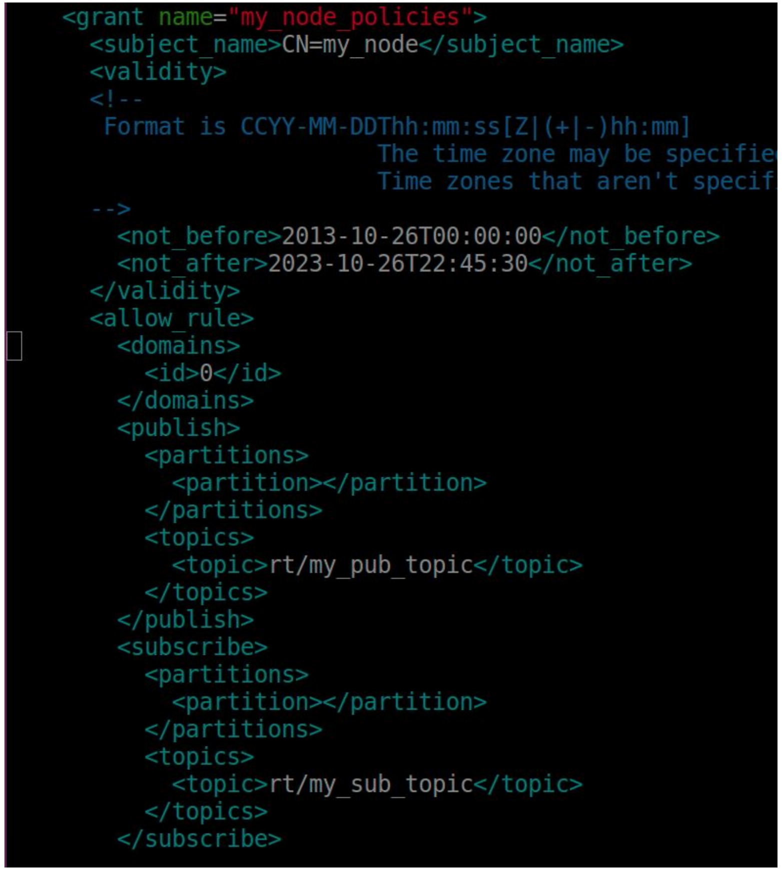
- Create policies files for a node:



```
nodes:
  my_node:
    topics:
      my_pub_topic:
        allow: p
      my_sub_topic:
        allow: s
```

How to generate SROS 2 artifacts?

- Create permission files for a nodes:
  - $ ros2 security create_permissions \
    my_keystore my_node
    ./my_node_policies.yaml



```
  <grant name="my_node_policies">
    <subject_name>CN=my_node</subject_name>
    <validity>
    <!--
     Format is CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]
                         The time zone may be specifie
                         Time zones that aren't specif
    -->
      <not_before>2013-10-26T00:00:00</not_before>
      <not_after>2023-10-26T22:45:30</not_after>
    </validity>
    <allow_rule>
      <domains>
        <id>0</id>
      </domains>
      <publish>
        <partitions>
          <partition></partition>
        </partitions>
        <topics>
          <topic>rt/my_pub_topic</topic>
        </topics>
      </publish>
      <subscribe>
        <partitions>
          <partition></partition>
        </partitions>
        <topics>
          <topic>rt/my_sub_topic</topic>
        </topics>
      </subscribe>
```

Let’s try it!

```
$ docker run -it --rm osrf/ros2:bouncy-desktop
# source /opt/ros/bouncy/setup.bash
# mkdir ~/my_ros2_ws && cd ~/my_ros2_ws
# ros2 security create_keystore demo_keys
# ros2 security create_key demo_keys talker
# ros2 security create_key demo_keys listener
```

Now let’s run our secure nodes:

```
# export ROS_SECURITY_ROOT_DIRECTORY=~/my_ros2_ws/demo_keys
# export ROS_SECURITY_ENABLE=true
# export ROS_SECURITY_STRATEGY=Enforce

# ros2 run demo_nodes_cpp talker &
# ros2 run demo_nodes_py listener
```

Let’s try it!

Access Control:

Create ~/my_ros2_ws/pub_sub_policies.yaml with:

```
nodes:
  listener:
    topics:
      chatter:
        allow: s # can subscribe to chatter
  talker:
    topics:
      chatter:
        allow: p # can publish on chatter
```

Let’s try it!

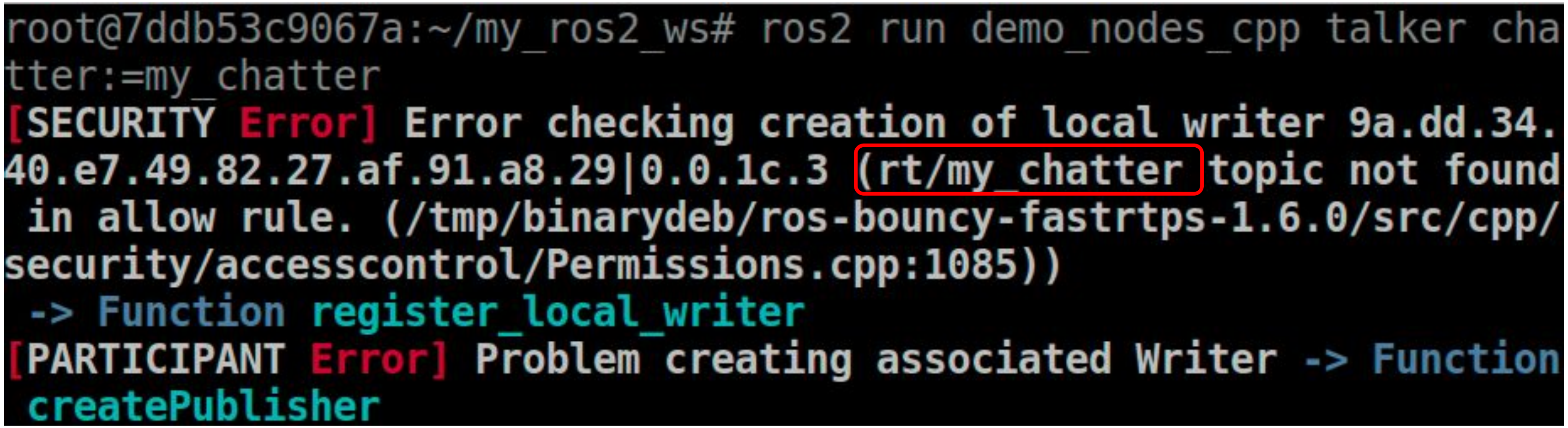
Create the permissions:

```
# ros2 security create_permission demo_keys talker pub_sub_policies.yaml
# ros2 security create_permission demo_keys listener pub_sub_policies.yaml


# ros2 run demo_nodes_cpp talker &
# ros2 run demo_nodes_py listener
```

Let’s try it!

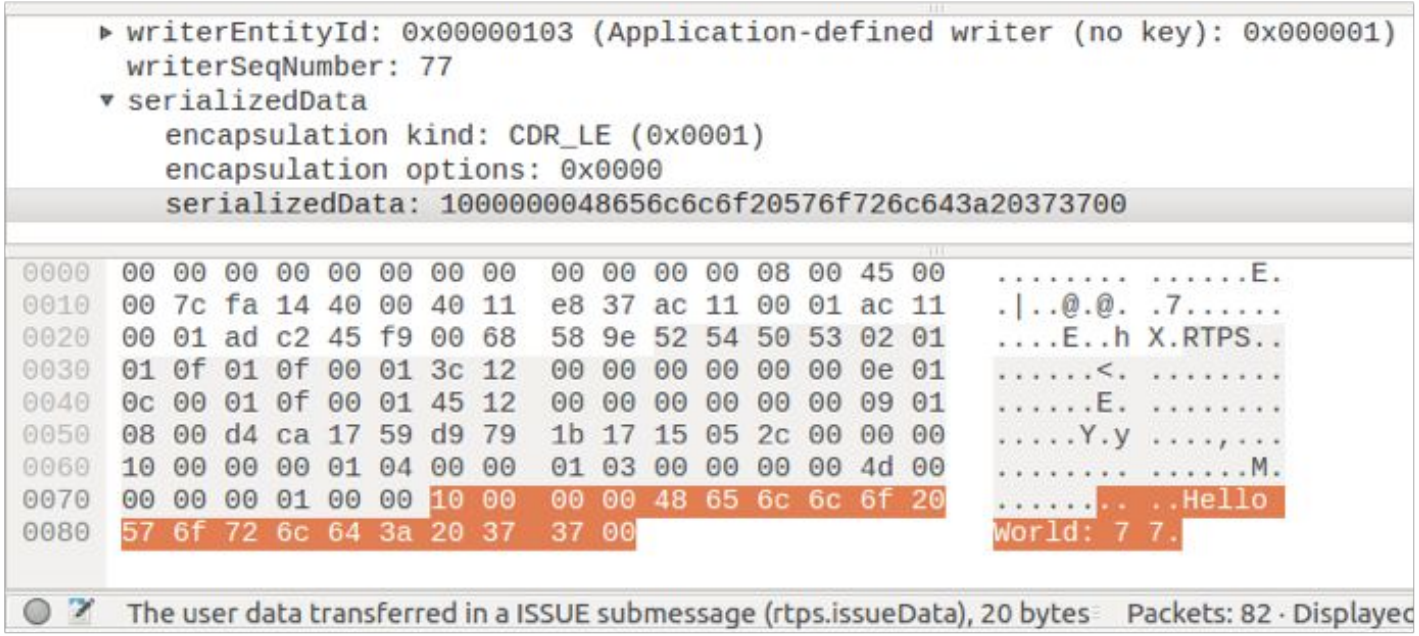
Let’s remap the topic on which talker publishes:

# ros2 run demo_nodes_cpp talker chatter:=my_chatter



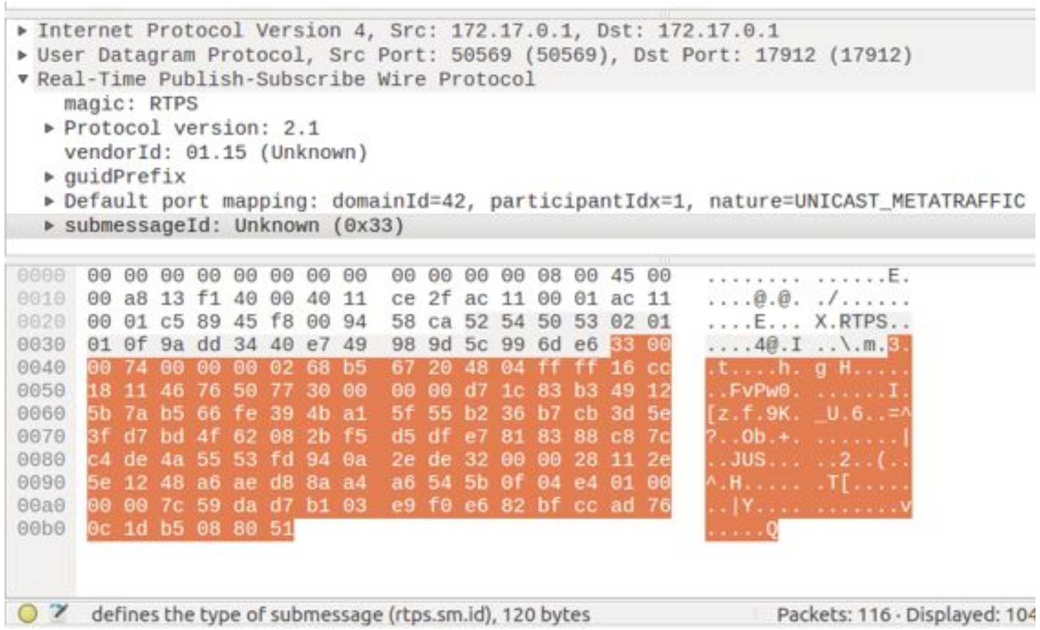
```
root@7ddb53c9067a:~/my_ros2_ws# ros2 run demo_nodes_cpp talker cha
tter:=my_chatter
[SECURITY Error] Error checking creation of local writer 9a.dd.34.
40.e7.49.82.27.af.91.a8.29|0.0.1c.3 (rt/my_chatter topic not found
 in allow rule. (/tmp/binarydeb/ros-bouncy-fastrtps-1.6.0/src/cpp/
security/accesscontrol/Permissions.cpp:1085))
 -> Function register_local_writer
[PARTICIPANT Error] Problem creating associated Writer -> Function
 createPublisher
```

Seeing in wireshark (clear text)



```
▸ writerEntityId: 0x00000103 (Application-defined writer (no key): 0x000001)
  writerSeqNumber: 77
▾ serializedData
    encapsulation kind: CDR_LE (0x0001)
    encapsulation options: 0x0000
    serializedData: 1000000048656c6c6f20576f726c643a20373700

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 7c fa 14 40 00 40 11  e8 37 ac 11 00 01 ac 11   .|..@.@. .7......
0020  00 01 ad c2 45 f9 00 68  58 9e 52 54 50 53 02 01   ....E..h X.RTPS..
0030  01 0f 01 0f 00 01 3c 12  00 00 00 00 00 00 0e 01   ......<. ........
0040  0c 00 01 0f 00 01 45 12  00 00 00 00 00 00 09 01   ......E. ........
0050  08 00 d4 ca 17 59 d9 79  1b 17 15 05 2c 00 00 00   .....Y.y ....,...
0060  10 00 00 00 01 04 00 00  01 03 00 00 00 00 4d 00   ........ ......M.
0070  00 00 00 01 00 00 10 00  00 00 48 65 6c 6c 6f 20   ........ ..Hello
0080  57 6f 72 6c 64 3a 20 37  37 00                     World: 7 7.

The user data transferred in a ISSUE submessage (rtps.issueData), 20 bytes    Packets: 82 · Displayed
```

Seeing in wireshark (encrypted)



```
▶ Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.1
▶ User Datagram Protocol, Src Port: 50569 (50569), Dst Port: 17912 (17912)
▼ Real-Time Publish-Subscribe Wire Protocol
    magic: RTPS
  ▶ Protocol version: 2.1
    vendorId: 01.15 (Unknown)
  ▶ guidPrefix
  ▶ Default port mapping: domainId=42, participantIdx=1, nature=UNICAST_METATRAFFIC
  ▶ submessageId: Unknown (0x33)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 a8 13 f1 40 00 40 11  ce 2f ac 11 00 01 ac 11   ....@.@. ./......
0020  00 01 c5 89 45 f8 00 94  58 ca 52 54 50 53 02 01   ....E... X.RTPS..
0030  01 0f 9a dd 34 40 e7 49  98 9d 5c 99 6d e6 33 00   ....4@.I ..\.m.3.
0040  00 74 00 00 00 02 68 b5  67 20 48 04 ff ff 16 cc   .t....h. g H.....
0050  18 11 46 76 50 77 30 00  00 00 d7 1c 83 b3 49 12   ..FvPw0. ......I.
0060  5b 7a b5 66 fe 39 4b a1  5f 55 b2 36 b7 cb 3d 5e   [z.f.9K. _U.6..=^
0070  3f d7 bd 4f 62 08 2b f5  d5 df e7 81 83 88 c8 7c   ?..Ob.+. .......|
0080  c4 de 4a 55 53 fd 94 0a  2e de 32 00 00 28 11 2e   ..JUS... ..2..(..
0090  5e 12 48 a6 ae d8 8a a4  a6 54 5b 0f 04 e4 01 00   ^.H..... .T[.....
00a0  00 00 7c 59 da d7 b1 03  e9 f0 e6 82 bf cc ad 76   ..|Y.... .......v
00b0  0c 1d b5 08 80 51                                  .....Q

defines the type of submessage (rtps.sm.id), 120 bytes          Packets: 116 · Displayed: 104
```