# Fast DDS Spy Documentation

***Release ..***

**eProsima**

**Apr 09, 2024**

# INTRODUCTION

*eProsima Fast DDS Spy* is CLI interactive tool that allows to introspect a DDS network in human readable format. It is possible to query the network about the DomainParticipants connected, their endpoints (DataWriters and DataReaders) and the topics they communicate in. It is also possible to see the user data sent through network topics in a schematic format in run time.

*eProsima Fast DDS Spy* is a tool that introspect or "sniffing" DDS packages in the network and maintain a local database that is accessible from a interactive CLI. *Fast DDS Spy* responds to user commands introduced by text and print in `stdin` the information requested. This tool has several commands to interact with, that allows to get information regarding the status of the network. It supports to list topics existing, list Participants, DataReaders, DataWriters and even read user data in real time in a human readable format.


*eProsima Fast DDS Spy* is easily configurable and installed with a default setup, so that DDS topics, data types and entities are automatically discovered without the need to specify the types of data. This is because this tool exploits the DynamicTypes functionality of eProsima Fast DDS, the C++ implementation of the DDS (Data Distribution Service) Specification defined by the Object Management Group (OMG).

CHAPTER

ONE

# CONTACTS AND COMMERCIAL SUPPORT

Find more about us at eProsima's webpage.

Support available at:

- Email: support@eprosima.com
- Phone: +34 91 804 34 48

CHAPTER

TWO

# CONTRIBUTING TO THE DOCUMENTATION

*Fast DDS Spy Documentation* is an open source project, and as such all contributions, both in the form of feedback and content generation, are most welcomed. To make such contributions, please refer to the Contribution Guidelines hosted in our GitHub repository.

CHAPTER

THREE

# STRUCTURE OF THE DOCUMENTATION

This documentation is organized into the sections below.

- *Installation Manual*
- *User Manual*
- *Developer Manual*
- *Release Notes*

*eProsima Fast DDS Spy* is CLI interactive tool that allows to introspect a DDS network in human readable format. It is possible to query the network about the DomainParticipants connected, their endpoints (DataWriters and DataReaders) and the topics they communicate in. It is also possible to see the user data sent through network topics in a schematic format in run time.

## 3.1 Overview

*eProsima Fast DDS Spy* is a tool that introspect or "sniffing" DDS packages in the network and maintain a local database that is accessible from a interactive CLI. *Fast DDS Spy* responds to user commands introduced by text and print in `stdin` the information requested. This tool has several commands to interact with, that allows to get information regarding the status of the network. It supports to list topics existing, list Participants, DataReaders, DataWriters and even read user data in real time in a human readable format.


*eProsima Fast DDS Spy* is easily configurable and installed with a default setup, so that DDS topics, data types and entities are automatically discovered without the need to specify the types of data. This is because this tool exploits the

DynamicTypes functionality of eProsima Fast DDS, the C++ implementation of the DDS (Data Distribution Service) Specification defined by the Object Management Group (OMG).

## 3.2 Contacts and Commercial support

Find more about us at eProsima's webpage.

Support available at:

- Email: support@eprosima.com
- Phone: +34 91 804 34 48

## 3.3 Contributing to the documentation

*Fast DDS Spy Documentation* is an open source project, and as such all contributions, both in the form of feedback and content generation, are most welcomed. To make such contributions, please refer to the Contribution Guidelines hosted in our GitHub repository.

## 3.4 Structure of the documentation

This documentation is organized into the sections below.

- *Installation Manual*
- *User Manual*
- *Developer Manual*
- *Release Notes*

## 3.5 Fast DDS Spy on Windows

**Warning:** The current version of *Fast DDS Spy* does not have installers for Windows platforms. Please refer to the *Windows installation from sources* section to learn how to build *Fast DDS Spy* on Windows from sources.

## 3.6 Fast DDS Spy on Linux

**Warning:** The current version of *Fast DDS Spy* does not have installers for Linux platforms. Please refer to the *Linux installation from sources* section to learn how to build *Fast DDS Spy* on Linux from sources.

## 3.7 Docker Image (recommended)

eProsima distributes a Docker image of *Fast DDS Spy* with Ubuntu 22.04 as base image. This image launches an instance of *Fast DDS Spy* that is configured using a *YAML* configuration file provided by the user and shared with the Docker container. The steps to run *Fast DDS Spy* in a Docker container are explained below.

1. Download the compressed Docker image in `.tar` format from the eProsima Downloads website. It is strongly recommended to download the image corresponding to the latest version of *Fast DDS Spy*.

2. Extract the image by executing the following command:

   ```
   load ubuntu-fastddsspy:<version>.tar
   ```

   where `version` is the downloaded version of *Fast DDS Spy*.

3. Build a *Fast DDS Spy* configuration YAML file on the local machine. This will be the *Fast DDS Spy* configuration file that runs inside the Docker container. Open your preferred text editor and copy a full configuration example into the `/<fastddsspy>/FASTDDSSPY_CONFIGURATION.yaml` file, where `fastddsspy` is the path where to execute the tool. To make this accessible from the Docker container we will create a shared volume containing just this file. This is explained in next point.

4. Run the Docker container executing the following command:

   ```
   docker run -it \
       --net=host \
       --ipc=host \
       --privileged \
       -v /<fastddsspy>/FASTDDSSPY_CONFIGURATION.yaml:/root/FASTDDSSPY_CONFIGURATION.
   ↪yaml \
       ubuntu-fastddsspy:v0.4.0
   ```

   It is important to mention that both the path to the configuration file hosted in the local machine and the one created in the Docker container must be absolute paths in order to share just one single file as a shared volume.

   After executing the previous command you should be able to see the initialization traces from the *Fast DDS Spy* running in the Docker container. If you want to terminate the application gracefully, just press `Ctrl+C` to stop the execution of *Fast DDS Spy*.

## 3.8 Example of usage

This example will serve as a hands-on tutorial, aimed at introducing some of the key concepts and features that *eProsima Fast DDS Spy* has to offer.

### 3.8.1 Prerequisites

It is required to have *eProsima Fast DDS Spy* previously installed using one of the following installation methods:

- *Fast DDS Spy on Windows*
- *Fast DDS Spy on Linux*
- *Docker Image (recommended)*

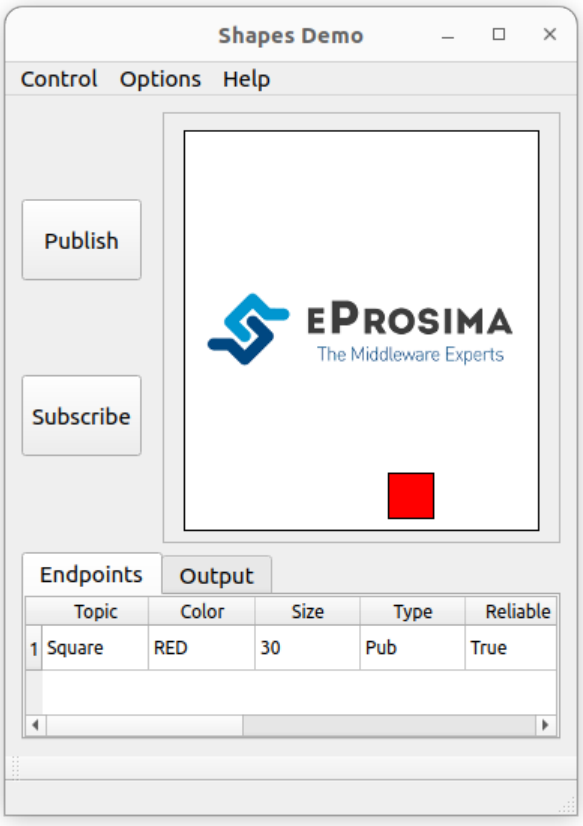
Additionally, ShapesDemo is required to publish and subscribe shapes of different colors and sizes. Install it by following any of the methods described in the given links:

- Windows installation from binaries

- Linux installation from sources
- Docker Image

### 3.8.2 Start ShapesDemo

Let us launch a ShapesDemo instance and start publishing in topics `Square` with default settings.




### 3.8.3 Spy configuration

*eProsima Fast DDS Spy* runs with default configuration settings.

Additionally, it is possible to change the default configuration parameters by means of a YAML configuration file.

**Note:** Please refer to *Configuration* for more information on how to configure a *eProsima Fast DDS Spy*.

### 3.8.4 Spy execution

Source the following file to setup the *eProsima Fast DDS Spy* environment:

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

Launch an *eProsima Fast DDS Spy* instance as executing the following command:

```
fastddsspy
```

Try out all the commands DDS Spy has to offer:

- participants

```
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1
- ...
```

- datawriters

```
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1
- ...
```

- topics

```
- name: Circle
  type: ShapeType
  datawriters: 2
  datareaders: 2
  rate: 13.0298 Hz
- name: Square
  type: ShapeType
  datawriters: 3
  datareaders: 2
  rate: 26.6975 Hz
```

- topics Square

```
name: Circle
type: ShapeType
datawriters:
  - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
datareaders:
  - 01.0f.44.59.c9.65.78.e5.00.00.00.00|0.0.2.7
rate: 13.0418 Hz
dynamic_type_discovered: true
```

- help

```
Insert a command for Fast DDS Spy:
>> help
Fast DDS Spy is an interactive CLI that allow to instrospect DDS networks.
Each command shows data related with the network in Yaml format.
Commands available and the information they show:
    help                    : this help.
    version                 : tool version.
    quit                    : exit interactive CLI and close program.
    participants            : DomainParticipants discovered in the network.
    participants verbose    : verbose information about DomainParticipants discovered in␣
↪the network.
    participants <Guid>     : verbose information related with a specific␣
↪DomainParticipant.
    writers                 : DataWriters discovered in the network.
    writers verbose         : verbose information about DataWriters discovered in the␣
↪network.
    writers <Guid>          : verbose information related with a specific DataWriter.
    reader                  : DataReaders discovered in the network.
    reader verbose          : verbose information about DataReaders discovered in the␣
↪network.
    reader <Guid>           : verbose information related with a specific DataReader.
    topics                  : Topics discovered in the network.
    topics verbose          : verbose information about Topics discovered in the network.
    topics <name>           : verbose information related with a specific Topic.
    show <name>             : data of a specific Topic (Data Type must be discovered).
    show <name> verbose     : data with additional source info of a specific Topic.
    show all                : verbose data of all topics (only those whose Data Type is␣
↪discovered).

Notes and comments:
    To exit from data printing, press enter.
    Each command is accessible by using its first letter (h/v/q/p/w/r/t/s).

For more information about these commands and formats, please refer to the documentation:
https://fast-dds-spy.readthedocs.io/en/latest/
```

Stop *eProsima Fast DDS Spy* typing `exit`.

### 3.8.5 Next Steps

Since the main steps for running Fast DDS Spy have already been explained, you can now continue by applying a configuration file to this tool to adjust it to your monitoring and debugging needs. These configurations include settings to enable or disable the DDS communication transports used by Fast DDS Spy, set the DDS Domain to monitor, or define lists of allowed and blocked topics, among others.

Please refer to the *Configuration* section of this documentation to know more about all settings available for the Fast DDS Spy.

## 3.9 User Interface

*eProsima Fast DDS Spy* is a *CLI* user application executed from command line and configured through a *YAML* configuration file.



### 3.9.1 Run application

Run *eProsima Fast DDS Spy* application by using command `fastddsspy`.

#### Source Dependency Libraries

*eProsima Fast DDS Spy* depends on some eProsima projects as *Fast DDS* or *DDS Pipe*. In order to correctly execute the application, make sure that these dependencies are properly sourced.

```
source <path-to-fastdds-installation>/install/setup.bash
```

**Note:** If Fast DDS has been installed in the system, these libraries would be sourced by default.

### 3.9.2 Application Arguments

The *eProsima Fast DDS Spy* application supports several input arguments:

| Command | Option | Long option | Value | Default Value |
|---|---|---|---|---|
| *Help Argument* | `-h` | `--help` | | |
| *Version Argument* | `-v` | `--version` | | |
| *Configuration File Argument* | `-c` | `--config-path` | Readable File Path | `./FASTDDSSPY_CONFIGURATION.yaml` |
| *Reload Time Argument* | `-r` | `--reload-time` | Period (in seconds) to reload configuration file | 0 |
| *Domain Argument* | | `--domain` | Domain to spy on | 0 |
| *Debug Argument* | `-d` | `--debug` | Debug mode | `false` |
| *Log Filter Argument* | | `--log-filter` | Filter the logs displayed | `FASTDDSSPY` |
| *Log Verbosity Argument* | | `--log-verbosity` | Maximum category of the logs displayed | `error` |

### Help Argument

It shows the usage information of the application.

```
Usage: Fast DDS Spy
Start an interactive CLI to introspect a DDS network.
General options:

Application help and information.
-h --help           Print this help message.
-v --version        Print version, branch and commit hash.

Application parameters
-c --config-path    Path to the Configuration File (yaml format) [Default: ./FASTDDSSPY_
↪CONFIGURATION.yaml].
-r --reload-time    Time period in seconds to reload configuration file. This is needed␣
↪when FileWatcher functionality is not available (e.g. config file is a symbolic link).␣
↪Value 0 does not reload file. [Default: 0].
   --domain         Set the domain (0-255) to spy on. [Default = 0].

Debug parameters
-d --debug          Set log verbosity to Info (Using this option with --log-filter and/
↪or --log-verbosity will head to undefined behaviour).
   --log-filter     Set a Regex Filter to filter by category the info and warning log␣
↪entries. [Default = "FASTDDSSPY"].
   --log-verbosity  Set a Log Verbosity Level higher or equal the one given. (Values␣
↪accepted: "info","warning","error" no Case Sensitive) [Default = "error"].
```

### Version Argument

It shows the current version of the *Fast DDS Spy* and the hash of the last commit of the compiled code.

### Configuration File Argument

A *Fast DDS Spy* supports *YAML* configuration file. Please refer to *Configuration* for more information on how to build this configuration file.

This *YAML* configuration can passed as argument to *Fast DDS Spy* when executed. If no configuration file is provided as argument, *Fast DDS Spy* will attempt to load a file named `FASTDDSSPY_CONFIGURATION.yaml` that must be in the same directory where the application is executed. If no configuration file is found, *Fast DDS Spy* will use *default configuration*.

### Reload Topics

This configuration file allows to allow and block DDS *Topics*. A modification in this file will modify the running application.

#### Reload Time Argument

This argument sets the time period in seconds to reload the configuration file.

#### Domain Argument

This argument sets the domain id of the *Fast DDS Spy*.

> **Warning:** If set, it will override the domain id set in the configuration file.

#### Debug Argument

This argument sets the log verbosity to Info.

> **Warning:** Using this option with *log filter* and/or *log verbosity* will head to undefined behaviour.

#### Log Filter Argument

Configure the *Fast DDS Spy* to print the logs that match the given filter. By default the filter is set to `FASTDDSSPY` for `warning` and `info` logs.

#### Log Verbosity Argument

Configure the *Fast DDS Spy* to print the logs up to a certain verbosity level. The verbosity levels are (from more to less restrictive): `error`, `warning`, and `info`.

### 3.9.3 Interactive application

The standard way to use this application is by running the *interactive CLI*. This is a user interface that repeatedly ask the user for a command, expecting a `stdin` command and arguments in order to retrieve the data querying the internal database.

```
Insert a command for Fast DDS Spy:
>>
```

Check the *following section* to see the available commands and their arguments, or use *help* command to get this information in `stdout`.

#### Close Application

Use *Quit* to stop and close the application.

### 3.9.4 One-shot application

*eProsima Fast DDS Spy* can be executed as a *one-shot* application. This is, an application that connects to a DDS network, queries a specific command to the internal database generated from such DDS network, and retrieves the information by `stdout`. Thus it will not ask the user for commands, neither wait before closing.

In order to execute *Fast DDS Spy* in *one-shot* mode add the command and arguments right after the last *tool argument*.

This kind of application depends on the network size and speed in order to discover every entity before showing the information and close. In order to configure the amount of time elapsed before querying for the information requested, use *Discovery Time*.

---

**Note:** Running a *Fast DDS Spy* instance will create DDS entities that will discover and connect everything in the same network. Thus, running *one-shot Fast DDS Spy* applications very frequently could affect to network performance, as each instance will be created and destroyed in the process.

---

#### Example

Let's assume we want to retrieve the information of every *DomainParticipant* running in a current network.

Using the *Interactive application*, this will be the process and the output:

```
<path>$ fastddsspy

 ____|              |        __ \   __ \    ___|        ___|
|     _` |   __|  __|      |   |  |   | \___ \      \___ \   __ \   |   |
__|  (   | \__ \  |        |   |  |   |       |           |  |   |  |   |
_|   \__,_| ____/ \__|     ____/  ____/  _____/      _____/   .__/  \__, |
                                                             _|     ____/


Insert a command for Fast DDS Spy:
>> participants
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1

Insert a command for Fast DDS Spy:
>> quit
<path>$
```

However, using *One-shot application* this will be expected result:

```
<path>$ fastddsspy participants
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1
<path>$
```

## 3.10 Configuration

A *Fast DDS Spy* instance can be configured by a *YAML* configuration file. In order to retrieve a configuration file to a *Fast DDS Spy*, use *Configuration File Argument*.

### 3.10.1 DDS Configurations

The YAML Configuration supports a `dds` **optional** tag that contains certain *DDS* configurations. The values available to configure are:

#### Topic Filtering

The *Fast DDS Spy* automatically detects the topics that are being used in a DDS Network. The *Fast DDS Spy* then creates internal DDS *Readers* for each topic to process the data published. The *Fast DDS Spy* allows filtering DDS *Topics*, that is, it allows users to configure the DDS *Topics* to process. These data filtering rules can be configured under the `allowlist` and `blocklist` tags. If the `allowlist` and `blocklist` are not configured, the *Fast DDS Spy* will process all the data published on the topics it discovers. If both the `allowlist` and `blocklist` are configured and a topic appears in both of them, the `blocklist` has priority and the topic will be blocked.

Topics are determined by the tags `name` (required) and `type`, both of which accept wildcard characters.

---

**Note:** Placing quotation marks around values in a YAML file is generally optional, but values containing wildcard characters do require single or double quotation marks.

---

Consider the following example:

```
allowlist:
  - name: AllowedTopic1
    type: Allowed

  - name: AllowedTopic2
    type: "*"

  - name: HelloWorldTopic
    type: HelloWorld

blocklist:
  - name: "*"
    type: HelloWorld
```

In this example, the data in the topic `AllowedTopic1` with type `Allowed` and the data in the topic `AllowedTopic2` with any type will be processed by the *Fast DDS Spy*. The data in the topic `HelloWorldTopic` with type `HelloWorld` will be blocked, since the `blocklist` is blocking all topics with any name and with type `HelloWorld`.

### Topic QoS

The following is the set of QoS that are configurable for a topic. For more information on topics, please read the Fast DDS Topic section.

| Quality of Service | Yaml tag | Data type | Default value | QoS set |
|---|---|---|---|---|
| Reliability | `reliability` | *bool* | `false` | `RELIABLE` / `BEST_EFFORT` |
| Durability | `durability` | *bool* | `false` | `TRANSIENT_LOCAL` / `VOLATILE` |
| Ownership | `ownership` | *bool* | `false` | `EXCLUSIVE_OWNERSHIP_QOS` / `SHARED_OWNERSHIP_QOS` |
| Partitions | `partitions` | *bool* | `false` | Topic with / without partitions |
| Key | `keyed` | *bool* | `false` | Topic with / without key |
| History Depth | `history-depth` | *unsigned integer* | `5000` | *History Depth* |
| Max Reception Rate | `max-rx-rate` | *float* | `0` (unlimited) | *Max Reception Rate* |
| Downsampling | `downsampling` | *unsigned integer* | `1` | *Downsampling* |

**Warning:** Manually configuring `TRANSIENT_LOCAL` durability may lead to incompatibility issues when the discovered reliability is `BEST_EFFORT`. Please ensure to always configure the `reliability` when configuring the `durability` to avoid the issue.

### History Depth

The `history-depth` tag configures the history depth of the Fast DDS internal entities. By default, the depth of every RTPS History instance is `5000`. Its value should be decreased when the sample size and/or number of created endpoints (increasing with the number of topics) are big enough to cause memory exhaustion issues.

### Max Reception Rate

The `max-rx-rate` tag limits the frequency [Hz] at which samples are processed by discarding messages received before `1/max-rx-rate` seconds have passed since the last processed message. It only accepts non-negative numbers. By default it is set to `0`; it processes samples at an unlimited reception rate.

### Downsampling

The `downsampling` tag reduces the sampling rate of the received data by only keeping *1* out of every *n* samples received (per topic), where *n* is the value specified under the `downsampling` tag. When the `max-rx-rate` tag is also set, downsampling only applies to messages that have passed the `max-rx-rate` filter. It only accepts positive integers. By default it is set to 1; it accepts every message.

### Manual Topics

A subset of *Topic QoS* can be manually configured for a specific topic under the tag `topics`. The tag `topics` has a required `name` tag that accepts wildcard characters. It also has two optional tags: a `type` tag that accepts wildcard characters and a `qos` tag with the *Topic QoS* that the user wants to manually configure. If a `qos` is not manually configured, it will get its value by discovery.

```
topics:
  - name: "temperature/*"
    type: "temperature/types/*"
    qos:
      max-tx-rate: 15
      downsampling: 2
```

**Note:** The *Topic QoS* configured in the Manual Topics take precedence over the *Specs Topic QoS*.

### DDS Domain Id

In order to execute a *Fast DDS Spy* instance in a *Domain Id* different than the default (`0`) use tag `domain`.

### Ignore Participant Flags

A set of discovery traffic filters can be defined in order to add an extra level of isolation. This configuration option can be set through the `ignore-participant-flags` tag:

```
ignore-participant-flags: no_filter                          # No filter (default)
# or
ignore-participant-flags: filter_different_host              # Discovery traffic from␣
↪another host is discarded
# or
ignore-participant-flags: filter_different_process           # Discovery traffic from␣
↪another process on same host is discarded
# or
ignore-participant-flags: filter_same_process                # Discovery traffic from␣
↪own process is discarded
# or
ignore-participant-flags: filter_different_and_same_process  # Discovery traffic from␣
↪own host is discarded
```

See Ignore Participant Flags for more information.

#### Custom Transport Descriptors

By default, *Fast DDS Spy* internal participants are created with enabled UDP and Shared Memory transport descriptors. The use of one or the other for communication will depend on the specific scenario, and whenever both are viable candidates, the most efficient one (Shared Memory Transport) is automatically selected. However, a user may desire to force the use of one of the two, which can be accomplished via the `transport` configuration tag.

```
transport: builtin    # UDP & SHM (default)
# or
transport: udp        # UDP only
# or
transport: shm        # SHM only
```

**Warning:** When configured with `transport: shm`, *Fast DDS Spy* will only communicate with applications using Shared Memory Transport exclusively (with disabled UDP transport).

#### Interface Whitelist

Optional tag `whitelist-interfaces` allows to limit the network interfaces used by UDP and TCP transport. This may be useful to only allow communication within the host (note: same can be done with *Ignore Participant Flags*). Example:

```
whitelist-interfaces:
  - "127.0.0.1"    # Localhost only
```

See Interface Whitelist for more information.

#### Topic type format

The optional `ros2-types` tag enables specification of the format for displaying schemas. When set to `true`, schemas are displayed in ROS 2 message format (.msg). If set to `false`, schemas are displayed in OMG IDL format (.idl).

### 3.10.2 Specs Configurations

The YAML Configuration supports a `specs` **optional** tag that contains certain options related with the overall configuration of the application. The values available to configure are:

#### Number of Threads

`specs` supports a `threads` **optional** value that allows the user to set a maximum number of threads for the internal `ThreadPool`. This ThreadPool allows to limit the number of threads spawned by the application. This improves the performance of the data transmission between participants.

This value should be set by each user depending on each system characteristics. By default, this value is `12`.

### Discovery Time

`specs` supports a `discovery-time` **optional** value that allows the user to set the time (in milliseconds) before a *One-shot application* retrieves the output and closes. This parameter is useful for very big networks, as *Fast DDS Spy* may not discover the whole network fast enough to return a complete information. By default, this value is `1000` (1 second).

### QoS

`specs` supports a `qos` **optional** tag to configure the default values of the *Topic QoS*.

---

**Note:** The *Topic QoS* configured in `specs` can be overwritten by the *Manual Topics*.

---

### Logging

`specs` supports a `logging` **optional** tag to configure the *Fast DDS Spy* logs. Under the `logging` tag, users can configure the type of logs to display and filter the logs based on their content and category. When configuring the verbosity to `info`, all types of logs, including informational messages, warnings, and errors, will be displayed. Conversely, setting it to `warning` will only show warnings and errors, while choosing `error` will exclusively display errors. By default, the filter allows all errors to be displayed, while selectively permitting warning and informational messages from `FASTDDSSPY` category.

---

**Note:** Configuring the logs via the Command-Line is still active and takes precedence over YAML configuration when both methods are used simultaneously.

---

| Logging | Yaml tag | Description | Data type | Default value | Possible values |
|---|---|---|---|---|---|
| Verbosity | `verbosity` | Show messages of equal or higher importance. | *enum* | `error` | `info` / `warning` / `error` |
| Filter | `filter` | Regex to filter the category or message of the logs. | *string* | info : `FASTDDSSPY` warning : `FASTDDSSPY` error : `""` | Regex string |

---

**Note:** For the logs to function properly, the `-DLOG_INFO=ON` compilation flag is required.

---

The *Fast DDS Spy* prints the logs by default (warnings and errors in the standard error and infos in the standard output). The *Fast DDS Spy*, however, can also publish the logs in a DDS topic. To publish the logs, under the tag `publish`, set `enable: true` and set a `domain` and a `topic-name`. The type of the logs published is defined as follows:

**LogEntry.idl**

```
const long UNDEFINED = 0x10000000;
const long SAMPLE_LOST = 0x10000001;
const long TOPIC_MISMATCH_TYPE = 0x10000002;
const long TOPIC_MISMATCH_QOS = 0x10000003;

enum Kind {
  Info,
```

(continues on next page)

(continued from previous page)

```
  Warning,
  Error
};

struct LogEntry {
  @key long event;
  Kind kind;
  string category;
  string message;
  string timestamp;
};
```

**Note:** The type of the logs can be published by setting `publish-type: true`.

**Example of usage**

```
logging:
  verbosity: info
  filter:
    error: "DDSPIPE|FASTDDSSPY"
    warning: "DDSPIPE|FASTDDSSPY"
    info: "FASTDDSSPY"
  publish:
    enable: true
    domain: 84
    topic-name: "FastDdsSpyLogs"
    publish-type: false
  stdout: true
```

### 3.10.3 General Example

A complete example of all the configurations described on this page can be found below.

**Warning:** This example can be used as a quick reference, but it may not be correct due to incompatibility or exclusive properties. **Do not take it as a working example**.

```
dds:
  domain: 0

  allowlist:
    - name: "topic_name"
      type: "topic_type"

  blocklist:
    - name: "topic_name"
      type: "topic_type"

  topics:
```

(continues on next page)

(continued from previous page)

```
    - name: "temperature/*"
      type: "temperature/types/*"
      qos:
        max-rx-rate: 5
        downsampling: 1

  ignore-participant-flags: no_filter
  transport: builtin
  whitelist-interfaces:
    - "127.0.0.1"

  ros2-types: false

specs:
  threads: 12
  discovery-time: 1000

  qos:
    history-depth: 5000
    max-rx-rate: 10
    downsampling: 2

  logging:
    verbosity: warning
    filter:
      error: ""
      warning: "FASTDDSSPY"
      info: "FASTDDSSPY"
    publish:
      enable: true
      domain: 0
      topic-name: "FastDdsSpyLogs"
      publish-type: false
    stdout: true
```

## 3.11 Commands

These are the commands supported so far by *Fast DDS Spy*. Every command explained in this section is available from the *Interactive application* and from the *One-shot application*. Most of the commands and arguments available have shortcuts and other names related. Thus, multiple words execute the same commands (e.g. *participant* = *participants* = *p*).

### 3.11.1 Entity Commands

The information retrieved by these commands follows *YAML* format and queries the application database about the network current status.

#### Participants

**Participants** is a command that retrieves information of the *DomainParticipants* currently active in the network.

#### Key-words

These are the key-words recognize as this command: `participant` `participants` `p` `P`.

#### Arguments

**Participants** command support 0 or 1 argument:

##### *No argument*

When no arguments are given to this command, the information shown is a **list** with every participant currently active in the network, giving their *Guid* and names. The output format is as follows: *Simple Participant info*.

#### Verbose

This argument queries for more complete information about each of the DomainParticipants in the network. It adds the information about endpoints and the topics they communicate in of each of the participants. The output got is a **list** of data with *verbose information*. Check the *verbose* section in order to know which key-words are available for this argument.

#### Guid

This argument requires a string with *Guid format*. This command queries the database for a **single DomainParticipant** and retrieves its *verbose information*. This Guid must exist inside the DDS network.

---

**Note:** If you are using *eProsima Fast DDS Spy* as one-shot application, you will need to put GUID in quotes.

---

#### Output Format

The participant information is retrieved in 2 formats depending on the verbose option.

#### Simple Participant info

```
name: <participant name>
guid: <guid>
```

#### Verbose Participant info

```
name: <participant name>
guid: <guid>
datawriters:
    - <topic name> [<topic data type name>] (<number of datawriters>)
    - ...
datareaders:
    - <topic name> [<topic data type name>] (<number of datareaders>)
    - ...
```

#### Example

Let's assume we have a DDS network where 2 ShapesDemo applications are running.

This would be the expected output for the command `participants`:

```
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1
- ...
```

This would be the expected output for the command `participants verbose`:

```
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.c1
  datawriters:
    - Triangle [ShapeType] (1)
  datareaders:
    - Square [ShapeType] (1)
- name: Fast DDS ShapesDemo Participant
  guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1
  datawriters:
    - Square [ShapeType] (2)
    - Circle [ShapeType] (1)
- ...
```

This would be the expected output for the command `participants 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1`:

```
name: Fast DDS ShapesDemo Participant
guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.c1
datawriters:
  - Square [ShapeType] (2)
  - Circle [ShapeType] (1)
```

### Writers

**Writers** is a command that retrieves information of the *DataWriters* currently active in the network.

### Key-words

These are the key-words recognize as this command: `writer` `writers` `datawriter` `datawriters` `publication` `publications` `w` `W`.

### Arguments

**Writers** command support 0 or 1 argument:

#### *No argument*

When no arguments are given to this command, the information shown is a **list** with every DataWriter currently active in the network, giving their :term:<Guid>, the name of their respective *DomainParticipants*, and the topic name and topic data type name. The output format is as follows: *Simple Writer info*.

### Verbose

This argument queries for more complete information about each of the DataWriters in the network. It adds information about the QoS. The output got is a **list** of DataWriters with *verbose information*. Check the *verbose* section in order to know which key-words are available for this argument.

### Guid

This argument requires a string with *Guid format*. This command queries the database for a **single DataWriter** and retrieves its *verbose information*. This Guid must exist inside the DDS network.

---

**Note:** If you are using *eProsima Fast DDS Spy* as one-shot application, you will need to put GUID in quotes.

---

### Output Format

The writer information is retrieved in 2 formats depending on the verbose option.

### Simple Writer info

```
guid: <guid>
name: <participant name>
topic: <topic name> [<topic data type name>]
```

### Verbose Writer info

```
name: <writer name>
guid: <guid>
topic:
    - <topic name>
    - <topic data type name>
qos:
    - durability: <volatile | transient-local>
    - reliability: <reliable | best-effort>
```

### Example

Let's assume we have a DDS network where 2 ShapesDemo applications are running.

This would be the expected output for the command `writers`:

```
- guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.2
  participant: Fast DDS ShapesDemo Participant
  topic: Triangle [ShapeType]
- guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
  participant: Fast DDS ShapesDemo Participant
  topic: Circle [ShapeType]
- ...
```

This would be the expected output for the command `writers verbose`:

```
- guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.2
  participant: Fast DDS ShapesDemo Participant
  topic:
    name: Triangle
    type: ShapeType
  qos:
    durability: volatile
    reliability: best-effort
- guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
  participant: Fast DDS ShapesDemo Participant
  topic:
    name: Circle
    type: ShapeType
  qos:
    durability: transient-local
    reliability: reliable
- ...
```

This would be the expected output for the command `writers 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2`:

```
guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
participant: Fast DDS ShapesDemo Participant
topic:
  name: Circle
```

(continues on next page)

(continued from previous page)

```
  type: ShapeType
qos:
  durability: transient-local
  reliability: reliable
```

### Readers

**Readers** is a command that retrieves information of the *DataReaders* currently active in the network.

### Key-words

These are the key-words recognize as this command: `reader` `readers` `datareader` `datareaders` `subscription` `subscriptions` `r` `R`.

### Arguments

**Readers** command support 0 or 1 argument:

#### *No argument*

When no arguments are given to this command, the information shown is a **list** with every DataReader currently active in the network, giving their *Guid*, the name of their respective *DomainParticipants*, and the topic name and topic data type name. The output format is as follows: *Simple Reader info*.

### Verbose

This argument queries for more complete information about each of the DataReaders in the network. It adds information about the QoS. The output got is a **list** of DataReaders with *verbose information*. Check the *verbose* section in order to know which key-words are available for this argument.

### Guid

This argument requires a string with *Guid format*. This command queries the database for a **single DataReader** and retrieves its *verbose information*. This Guid must exist inside the DDS network.

---

**Note:** If you are using *eProsima Fast DDS Spy* as one-shot application, you will need to put GUID in quotes.

---

### Output Format

The reader information is retrieved in 2 formats depending on the verbose option.

### Simple Reader info

```
guid: <guid>
name: <participant name>
topic: <topic name> [<topic data type name>]
```

### Verbose Reader info

```
name: <reader name>
guid: <guid>
topic:
    - <topic name>
    - <topic data type name>
qos:
    - durability: <volatile | transient-local>
    - reliability: <reliable | best-effort>
```

### Example

Let's assume we have a DDS network where 2 ShapesDemo applications are running.

This would be the expected output for the command `readers`:

```
- guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.2
  participant: Fast DDS ShapesDemo Participant
  topic: Triangle [ShapeType]
- guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
  participant: Fast DDS ShapesDemo Participant
  topic: Circle [ShapeType]
- ...
```

This would be the expected output for the command `readers verbose`:

```
- guid: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.2
  participant: Fast DDS ShapesDemo Participant
  topic:
    name: Triangle
    type: ShapeType
  qos:
    durability: volatile
    reliability: best-effort
- guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
  participant: Fast DDS ShapesDemo Participant
  topic:
    name: Circle
```

(continues on next page)

(continued from previous page)

```
    type: ShapeType
  qos:
    durability: transient-local
    reliability: reliable
- ...
```

This would be the expected output for the command `readers 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2`:

```
guid: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
participant: Fast DDS ShapesDemo Participant
topic:
  name: Circle
  type: ShapeType
qos:
  durability: transient-local
  reliability: reliable
```

### Topic

**Topic** is a command that retrieves information of the *Topics* with at least one endpoint currently active in the network.

### Key-words

These are the key-words recognize as this command: `topic` `topics` `t` `T`.

### Arguments

**Topic** command support 0 or 1 argument:

#### *No argument*

When no arguments are given to this command, the information shown is a **list** with every topic with at least one endpoint currently active in the network. The information shown is the topic name, data type name, number of writers and readers and the subscription rate measured in samples per second. The output format is as follows: *Simple Writer info*.

### Verbose

This argument queries for more complete information about each of the topics in the network. It adds the Guid of each endpoint on the topic and the whether the type has been discovered. The output is a **list** of data with *verbose information*. Check the *verbose* section in order to know which key-words are available for this argument.

#### Topic name

This argument requires a string with the topic name. This command queries the database for a **single Topic** and retrieves its *verbose information*. This Guid must exist inside the DDS network.

---

**Note:** If there are 2 topics with the same name and different Topic Data Type, only one of them could be visible. These is a circumstance that *DDS* allows, but it is strongly suggested not to do.

---

#### Output Format

The topic information is retrieved in 2 formats depending on the verbose option.

#### Simple Writer info

```
name: <topic name>
type: <data type name>
datawriters: <number of datawriters currently active>
datareaders: <number of datareaders currently active>
rate: <samples per second> Hz
```

#### Verbose Writer info

```
name: <topic name>
type: <data type name>
datawriters:
    - <Guid>
    - ...
datareaders:
    - <Guid>
    - ...
rate: <samples per second> Hz
dynamic_type_discovered: <bool>
```

#### Example

Let's assume we have a DDS network where 2 ShapesDemo applications are running.

This would be the expected output for the command `topics`:

```
- name: Circle
  type: ShapeType
  datawriters: 2
  datareaders: 2
  rate: 13.0298 Hz
- name: Square
  type: ShapeType
```

(continues on next page)

(continued from previous page)

```
  datawriters: 3
  datareaders: 2
  rate: 26.6975 Hz
```

This would be the expected output for the command `topics verbose`:

```
- name: Circle
  type: ShapeType
  datawriters:
    - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.3.2
  datareaders:
    - 01.0f.44.59.c9.65.78.e5.00.00.00.00|0.0.2.7
  rate: 13.0286 Hz
  dynamic_type_discovered: true
- name: Square
  type: ShapeType
  datawriters:
    - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.1.2
    - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.2.2
  datareaders:
    - 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.2.7
    - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.4.7
  rate: 26.685 Hz
  dynamic_type_discovered: true
```

This would be the expected output for the command `topics Square`:

```
name: Circle
type: ShapeType
datawriters:
  - 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
datareaders:
  - 01.0f.44.59.c9.65.78.e5.00.00.00.00|0.0.2.7
rate: 13.0418 Hz
dynamic_type_discovered: true
```

### 3.11.2 Data commands

This commands show user data being received by the application in real time.

#### Show

This command prints every User Data received in a human readable way. The information shown with this command is real-time data that is being received by *Fast DDS Spy*. In order to stop the command, press enter and the CLI will finish showing the data received.

**Note:** This is a real-time command that will not stop until enter is pressed.

#### Key-words

These are the key-words recognize as this command: `show` `print` `s` `S`.

#### Data Type discovered

In order for a *Topic* to be printable by the application, the *Fast DDS Spy* requires to know the data type of such topic. The information whether the topic data type is already discovered be the application can be shown by using *Topic* command.

#### Arguments

**Show** command support different combination of arguments:

#### Topic name

When a topic name is given, the information shown is the data received in real-time in the topic specified. The output format is as follows: *Simple Data format*.

#### Topic name + Verbose

Giving a topic name and the the *verbose argument* the output is the data received in real-time with additional meta-information as the topic name, the source timestamp, and the source *DataWriter Guid*. Data is printing using *Verbose Data format*.

#### All

This argument prints all topics which Data Type has been discovered. Data is printing using *Verbose Data format*.

#### Output Format

**Note:** The format of the data printed is not YAML. The correct YAML format will come in future releases.

The data information is retrieved in 2 formats depending on the verbose option.

#### Simple Data format

Only shows the data, by

```
---
<field name>: <value>
...
---
```

### Verbose Data format

```
topic: <topic name> [<topic data type name>]
data: <Guid>
timestamp: <YYYY/MM/DD hh:mm:ss>
data:
---
<field name>: <value>
...
---
```

### Example

Let's assume we have a DDS network where 2 ShapesDemo applications are running.

This would be the expected output for the command `show Circle`:

```
---
color: GREEN
x: 168
y: 125
shapesize: 30
---

---
color: GREEN
x: 173
y: 121
shapesize: 30
---

...
```

This would be the expected output for the command `show Circle verbose`:

```
topic: Circle [ShapeType]
data: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
timestamp: 2023/03/27 09:35:23
data:
---
color: GREEN
x: 72
y: 125
shapesize: 30
---

topic: Circle [ShapeType]
data: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.6.2
timestamp: 2023/03/27 09:35:24
data:
---
color: GREEN
```

(continues on next page)

(continued from previous page)

```
x: 66
y: 120
shapesize: 30
---

...
```

This would be the expected output for the command `datas 01.0f.22.ba.3b.47.ab.3c.00.00.00.00|0.0.1.c1`:

```
topic: Square [ShapeType]
data: 01.0f.44.59.da.57.de.ec.00.00.00.00|0.0.2.2
timestamp: 2023/03/27 09:35:25
data:
---
color: BLUE
x: 158
y: 70
shapesize: 20
---

topic: Triangle [ShapeType]
data: 01.0f.44.59.21.58.14.d2.00.00.00.00|0.0.1.2
timestamp: 2023/03/27 09:35:26
data:
---
color: YELLOE
x: 93
y: 75
shapesize: 25
---

...
```

### 3.11.3 Extra commands

These are other commands available in the application.

#### Help

Show the help information with all the commands available by *Fast DDS Spy*.

#### Key-words

These are the key-words recognize as this command: `h` `help` `H` `man`.

#### Version

Show the version information and commit of the application running.

#### Key-words

These are the key-words recognize as this command: `version` `v` `V`.

#### Quit

Stop and close the application.

#### Key-words

These are the key-words recognize as this command: `q` `x` `quit` `quit()` `exit` `exit()`.

### 3.11.4 Input format

Input format for the application command arguments.

#### Verbose

`verbose` is a common argument for most of the *Commands* available. This argument generate more complete and complex output data for the query executed by the command. In order to see verbose information, add one of the key-words right after the command and its required arguments.

#### Key-words

These are the key-words recognize as this argument: `verbose` `v` `-v` `--v` `V`.

#### All

Some commands support an `all` argument.

#### Key-words

These are the key-words recognize as this argument: `all` `a` `-a` `--a` `A`.

#### Guid

*Guid* identifies a DDS entity by a unique Id.

#### Format

The format for the Guid is a string of 12 hexadecimal numbers separated by `.`, the *Guid Prefix*, followed by `|` and then 4 more hexadecimal values representing the *Entity Id*. e.g. `01.0f.22.ba.3b.47.ab.3c.00.00.00.00|00.00.01.c1`.

### 3.11.5 Summary

| Command | Description | Arguments | KeyWords |
|---|---|---|---|
| *Participants* | Show DomainParticipant info | _ `verbose` <Guid> | `participant` `participants` `p` `P` |
| *Writers* | Show DataWriter info | _ `verbose` <Guid> | `writer` `writers` `datawriter` `datawriters` `publication` `publications` `w` `W` |
| *Readers* | Show DataReader info | _ `verbose` <Guid> | `reader` `readers` `datareader` `datareaders` `subscription` `subscriptions` `r` `R` |
| *Topic* | Show Topic info | _ `verbose` <topic name> | `topic` `topics` `t` `T` |
| *Show* | Show real-time receiving user data. | <topic name> <topic name> `verbose` `all` | `show` `print` `s` `S` |
| *Help* | Show help. | | `help` `man` `h` `H` |
| *Version* | Show version information. | | `version` `v` `V` |
| *Quit* | Stop and close application. | | `quit` `exit` `quit()` `exit()` `q` `Q` `x` |

## 3.12 Linux installation from sources

The instructions for installing the *eProsima Fast DDS Spy* from sources and its required dependencies are provided in this page. It is organized as follows:

### 3.12.1 Dependencies installation

*Fast DDS Spy* depends on *eProsima Fast DDS* library and certain Debian packages. This section describes the instructions for installing *Fast DDS Spy* dependencies and requirements in a Linux environment from sources. The following packages will be installed:

- `foonathan_memory_vendor`, an STL compatible C++ memory allocation library.
- `fastcdr`, a C++ library that serializes according to the standard CDR serialization mechanism.
- `fastrtps`, the core library of eProsima Fast DDS library.
- `cmake_utils`, an eProsima utils library for CMake.
- `cpp_utils`, an eProsima utils library for C++.
- `ddspipe`, an eProsima internal library that enables the communication of DDS interfaces.

First of all, the *Requirements* and *Dependencies* detailed below need to be met. Afterwards, the user can choose whether to follow either the *colcon* or the CMake installation instructions.

#### Requirements

The installation of *eProsima Fast DDS Spy* in a Linux environment from sources requires the following tools to be installed in the system:

- *CMake, g++, pip, wget and git*
- *Colcon* [optional]
- *Gtest* [for test only]

#### CMake, g++, pip, wget and git

These packages provide the tools required to install *eProsima Fast DDS Spy* and its dependencies from command line. Install CMake, g++, pip, wget and git using the package manager of the appropriate Linux distribution. For example, on Ubuntu use the command:

```
sudo apt install cmake g++ pip wget git
```

### Colcon

colcon is a command line tool based on CMake aimed at building sets of software packages. Install the ROS 2 development tools (colcon and vcstool) by executing the following command:

```
pip3 install -U colcon-common-extensions vcstool
```

**Note:** If this fails due to an Environment Error, add the `--user` flag to the `pip3` installation command.

### Gtest

Gtest is a unit testing library for C++. By default, *eProsima Fast DDS Spy* does not compile tests. It is possible to activate them with the opportune CMake options when calling colcon or CMake. For more details, please refer to the *CMake options* section. For a detailed description of the Gtest installation process, please refer to the Gtest Installation Guide.

It is also possible to clone the Gtest Github repository into the *eProsima Fast DDS Spy* workspace and compile it with colcon as a dependency package. Use the following command to download the code:

```
git clone --branch release-1.11.0 https://github.com/google/googletest src/googletest-
↪distribution
```

### Dependencies

*eProsima Fast DDS Spy* has the following dependencies, when installed from sources in a Linux environment:

- *Asio and TinyXML2 libraries*
- *OpenSSL*
- *yaml-cpp*
- *eProsima dependencies*

### Asio and TinyXML2 libraries

Asio is a cross-platform C++ library for network and low-level I/O programming, which provides a consistent asynchronous model. TinyXML2 is a simple, small and efficient C++ XML parser. Install these libraries using the package manager of the appropriate Linux distribution. For example, on Ubuntu use the command:

```
sudo apt install libasio-dev libtinyxml2-dev
```

#### OpenSSL

OpenSSL is a robust toolkit for the TLS and SSL protocols and a general-purpose cryptography library. Install OpenSSL using the package manager of the appropriate Linux distribution. For example, on Ubuntu use the command:

```
sudo apt install libssl-dev
```

#### yaml-cpp

yaml-cpp is a YAML parser and emitter in C++ matching the YAML 1.2 spec, and is used by *Fast DDS Spy* application to parse the provided configuration files. Install yaml-cpp using the package manager of the appropriate Linux distribution. For example, on Ubuntu use the command:

```
sudo apt install libyaml-cpp-dev
```

#### eProsima dependencies

If it already exists in the system an installation of *Fast DDS* and *DDS Pipe* libraries, just source this libraries when building *eProsima Fast DDS Spy* by running the following commands. In other case, just skip this step.

```
source <fastdds-installation-path>/install/setup.bash
source <ddspipe-installation-path>/install/setup.bash
```

### 3.12.2 Colcon installation

1. Create a `fastdds-spy` directory and download the `.repos` file that will be used to install *eProsima Fast DDS Spy* and its dependencies:

   ```
   mkdir -p ~/fastdds-spy/src
   cd ~/fastdds-spy
   wget https://raw.githubusercontent.com/eProsima/Fast-DDS-Spy/main/fastddsspy.repos
   vcs import src < fastddsspy.repos
   ```

   **Note:** In case there is already a *Fast DDS* installation in the system it is not required to download and build every dependency in the `.repos` file. It is just needed to download and build the *eProsima Fast DDS Spy* project having sourced its dependencies. Refer to section *eProsima dependencies* in order to check how to source *Fast DDS* library.

2. Build the packages:

   ```
   colcon build --packages-up-to-regex fastddsspy
   ```

**Note:** Being based on CMake, it is possible to pass the CMake configuration options to the `colcon build` command. For more information on the specific syntax, please refer to the CMake specific arguments page of the colcon manual.

### 3.12.3 CMake installation

There exist the possibility to install *Fast DDS Spy* by CMake, and could be see in following section. However *Colcon installation* is recommended.

### 3.12.4 Run an application

To run the *Fast DDS Spy* tool, source the installation path and execute the executable file that has been installed in `<install-path>/fastddsspy_tool/bin/fastddsspy`:

```
# If built has been done using colcon, all projects could be sourced as follows
source install/setup.bash
fastddsspy
```

Be sure that this executable has execution permissions.

### 3.12.5 Run tests

## 3.13 Windows installation from sources

The instructions for installing the *eProsima Fast DDS Spy* application from sources and its required dependencies are provided in this page. It is organized as follows:



### 3.13.1 Dependencies installation

*eProsima Fast DDS Spy* depends on *eProsima Fast DDS* library and certain Debian packages. This section describes the instructions for installing *eProsima Fast DDS Spy* dependencies and requirements in a Windows environment from sources. The following packages will be installed:

- `foonathan_memory_vendor`, an STL compatible C++ memory allocation library.
- `fastcdr`, a C++ library that serializes according to the standard CDR serialization mechanism.
- `fastrtps`, the core library of eProsima Fast DDS library.
- `cmake_utils`, an eProsima utils library for CMake.
- `cpp_utils`, an eProsima utils library for C++.
- `ddspipe`, an eProsima internal library that enables the communication of DDS interfaces.

First of all, the *Requirements* and *Dependencies* detailed below need to be met. Afterwards, the user can choose whether to follow either the *colcon* or the CMake installation instructions.

### Requirements

The installation of *eProsima Fast DDS* in a Windows environment from sources requires the following tools to be installed in the system:

- *Visual Studio*
- *Chocolatey*
- *CMake, pip3, wget and git*
- *Colcon* [optional]
- *Gtest* [for test only]

### Visual Studio

Visual Studio is required to have a C++ compiler in the system. For this purpose, make sure to check the `Desktop development with C++` option during the Visual Studio installation process.

If Visual Studio is already installed but the Visual C++ Redistributable packages are not, open Visual Studio and go to `Tools -> Get Tools and Features` and in the `Workloads` tab enable `Desktop development with C++`. Finally, click `Modify` at the bottom right.

### Chocolatey

Chocolatey is a Windows package manager. It is needed to install some of *eProsima Fast DDS*'s dependencies. Download and install it directly from the website.

### CMake, pip3, wget and git

These packages provide the tools required to install *eProsima Fast DDS* and its dependencies from command line. Download and install CMake, pip3, wget and git by following the instructions detailed in the respective websites. Once installed, add the path to the executables to the `PATH` from the *Edit the system environment variables* control panel.

### Colcon

colcon is a command line tool based on CMake aimed at building sets of software packages. Install the ROS 2 development tools (colcon and vcstool) by executing the following command:

```
pip3 install -U colcon-common-extensions vcstool
```

**Note:** If this fails due to an Environment Error, add the `--user` flag to the `pip3` installation command.

### Gtest

Gtest is a unit testing library for C++. By default, *eProsima Fast DDS Spy* does not compile tests. It is possible to activate them with the opportune CMake options when calling colcon or CMake. For more details, please refer to the *CMake options* section.

Run the following commands on your workspace to install Gtest.

```
git clone https://github.com/google/googletest.git
cmake -DCMAKE_INSTALL_PREFIX='C:\Program Files\gtest' -Dgtest_force_shared_crt=ON -
↪DBUILD_GMOCK=ON ^
    -B build\gtest -A x64 -T host=x64 googletest
cmake --build build\gtest --config Release --target install
```

or refer to the Gtest Installation Guide for a detailed description of the Gtest installation process.

### Dependencies

*eProsima Fast DDS Spy* has the following dependencies, when installed from sources in a Windows environment:

- *Asio and TinyXML2 libraries*
- *OpenSSL*
- *yaml-cpp*
- *eProsima dependencies*

### Asio and TinyXML2 libraries

Asio is a cross-platform C++ library for network and low-level I/O programming, which provides a consistent asynchronous model. TinyXML2 is a simple, small and efficient C++ XML parser. They can be downloaded directly from the links below:

- Asio
- TinyXML2

After downloading these packages, open an administrative shell with *PowerShell* and execute the following command:

```
choco install -y -s <PATH_TO_DOWNLOADS> asio tinyxml2
```

where <PATH_TO_DOWNLOADS> is the folder into which the packages have been downloaded.

### OpenSSL

OpenSSL is a robust toolkit for the TLS and SSL protocols and a general-purpose cryptography library. Download and install the latest OpenSSL version for Windows at this link. After installing, add the environment variable OPENSSL_ROOT_DIR pointing to the installation root directory.

For example:

```
OPENSSL_ROOT_DIR=C:\Program Files\OpenSSL-Win64
```

#### yaml-cpp

`yaml-cpp` is a YAML parser and emitter in C++ matching the YAML 1.2 spec, and is used by *Fast DDS Spy* application to parse the provided configuration files. From an administrative shell with *PowerShell*, execute the following commands in order to download and install `yaml-cpp` for Windows:

```
git clone --branch yaml-cpp-0.7.0 https://github.com/jbeder/yaml-cpp
cmake -DCMAKE_INSTALL_PREFIX='C:\Program Files\yamlcpp' -B build\yamlcpp yaml-cpp
cmake --build build\yamlcpp --target install    # If building in Debug mode, add --
↪config Debug
```

#### eProsima dependencies

If it already exists in the system an installation of *Fast DDS* and *DDS Pipe* libraries, just source this libraries when building the *eProsima Fast DDS Spy* application by using the command:

```
source <fastdds-installation-path>/install/setup.bash
source <ddspipe-installation-path>/install/setup.bash
```

In other case, just skip this step.

### 3.13.2 Colcon installation (recommended)

**Important:** Run colcon within a Visual Studio prompt. To do so, launch a *Developer Command Prompt* from the search engine.

1. Create a `Fast-DDS-Spy` directory and download the `.repos` file that will be used to install *eProsima Fast DDS Spy* and its dependencies:

   ```
   mkdir <path\to\user\workspace>\Fast-DDS-Spy
   cd <path\to\user\workspace>\Fast-DDS-Spy
   mkdir src
   wget https://raw.githubusercontent.com/eProsima/Fast-DDS-Spy/main/fastddsspy.repos
   vcs import src < fastddsspy.repos
   ```

   **Note:** In case there is already a *Fast DDS* installation in the system it is not required to download and build every dependency in the `.repos` file. It is just needed to download and build the *eProsima Fast DDS Spy* project having sourced its dependencies. Refer to section *eProsima dependencies* in order to check how to source *Fast DDS* library.

2. Build the packages:

   ```
   colcon build --packages-up-to-regex fastddsspy
   ```

**Note:** Being based on CMake, it is possible to pass the CMake configuration options to the `colcon build` command. For more information on the specific syntax, please refer to the CMake specific arguments page of the colcon manual.

### 3.13.3 CMake installation

There exist the possibility to install *Fast DDS Spy* by CMake, and could be see in following section. However *Colcon installation* is recommended.

### 3.13.4 Run an application

If the *eProsima Fast DDS Spy* was compiled using colcon, when running an instance of a *Fast DDS Spy*, the colcon overlay built in the dedicated `fastdds-spy` directory must be sourced. There are two possibilities:

- Every time a new shell is opened, prepare the environment locally by typing the command:

```
setup.bat
fastddsspy
```

- Add the sourcing of the colcon overlay permanently, by opening the *Edit the system environment variables* control panel, and adding the installation path to the `PATH`.

However, when running an instance of a *Fast DDS Spy* compiled using CMake, it must be linked with its dependencies where the packages have been installed. This can be done by opening the *Edit system environment variables* control panel and adding to the `PATH` the *eProsima Fast DDS Spy*, *Fast DDS*, *Fast CDR*, *DDS Pipe* installation directories:

- *Fast DDS*: `C:\\Program Files\\fastrtps`
- *Fast CDR*: `C:\\Program Files\\fastcdr`
- *DDS Pipe*: `C:\\Program Files\\ddspipe`
- *eProsima Fast DDS Spy*: `C:\\Program Files\\ddsrecord`

## 3.14 CMake options

*eProsima Fast DDS Spy* provides numerous CMake options for changing the behavior and configuration of *eProsima Fast DDS Spy*. These options allow the developer to enable/disable certain *eProsima Fast DDS Spy* settings by defining these options to `ON`/`OFF` at the CMake execution, or set the required path to certain dependencies.

**Warning:** These options are only for developers who installed *eProsima Fast DDS Spy* following the compilation steps described in *Linux installation from sources*.

| Option | Description | Possible values | Default |
|---|---|---|---|
| `CMAKE_BUILD_TYPE` | CMake optimization build type. | `Release` `Debug` | `Release` |
| `BUILD_DOCS` | Build the *eProsima Fast DDS Spy* documentation. | `OFF` `ON` | `OFF` |
| `BUILD_TESTS` | Build the *eProsima Fast DDS Spy* tools and documentation tests. | `OFF` `ON` | `OFF` |
| `LOG_INFO` | Activate *eProsima Fast DDS Spy* logs. It is set to `ON` if `CMAKE_BUILD_TYPE` is set to `Debug`. | `OFF` `ON` | `ON` if `Debug` `OFF` otherwise |
| `ASAN_BUILD` | Activate address sanitizer build. | `OFF` `ON` | `OFF` |
| `TSAN_BUILD` | Activate thread sanitizer build. | `OFF` `ON` | `OFF` |

## 3.15 Forthcoming Version

This release includes the following **Configuration Features**:

- New *domain argument* to configure the `domain` through the command-line.

## 3.16 Version v0.4.0

This release includes the following **Configuration Features**:

- New configuration option `logging` to configure the *Logs*.

This release includes the following **Documentation Updates**:

- Add a new *Logging* section to the *Configuration* page.

This release includes the following **Dependencies Update**:

| | Repository | Old Version | New Version |
|---|---|---|---|
| Foonathan Memory Vendor | eProsima/foonathan_memory_vendor | v1.3.1 | v1.3.1 |
| Fast CDR | eProsima/Fast-CDR | v2.1.3 | v2.2.0 |
| Fast DDS | eProsima/Fast-DDS | v2.13.1 | v2.14.0 |
| Dev Utils | eProsima/dev-utils | v0.5.0 | v0.6.0 |
| DDS Pipe | eProsima/DDS-Pipe | v0.3.0 | v0.4.0 |

## 3.17 Previous Versions

### 3.17.1 Version v0.3.0

This release includes the following **Configuration Features**:

- Display data type information in either ROS 2 format (MSG) or DDS format (IDL).
- *Topic configuration*.
- *Max Reception Rate* per topic.
- *Downsampling* per topic.
- Remove the support for built-in Topics.

This release includes the following **Internal Changes** and **Bugfixes**:

- Support both fastcdr v1 and v2.
- Add support for Fast DDS versions lower than v2.13.
- Set *app_id* and *app_metadata* attributes in *eProsima Fast DDS Spy* participants.
- Add participant and endpoint info topics to allowlist.
- Add internal builtin topics as DDS Topics.
- Reload internal topics when reloading the configuration file.
- Process an empty YAML when a configuration file isn't provided.

This release includes the following **Documentation Updates**:

- Update next steps in main example and remove warning in docker installation.
- Fix documentation build for Sphinx versions lower than 5.0.

This release includes the following **Dependencies Update**:

| | Repository | Old Version | New Version |
|---|---|---|---|
| Foonathan Memory Vendor | eProsima/foonathan_memory_vendor | v1.3.1 | v1.3.1 |
| Fast CDR | eProsima/Fast-CDR | v1.1.0 | v2.1.3 |
| Fast DDS | eProsima/Fast-DDS | v2.11.0 | v2.13.1 |
| Dev Utils | eProsima/dev-utils | v0.4.0 | v0.5.0 |
| DDS Pipe | eProsima/DDS-Pipe | v0.2.0 | v0.3.0 |

### 3.17.2 Version v0.2.0

This release includes the following **Configuration Features**:

- Support *Interface Whitelisting*.
- Support *Custom Transport Descriptors* (UDP or Shared Memory only).
- Support *Ignore Participant Flags*.

This release includes the following **Internal Changes** and **Bugfixes**:

- Print only active participants when using verbose mode in `participants` command.
- Fix C++ *namespace* ambiguities in yaml configuration.
- Change default log-filter to only show warning coming from DDS Spy source code.
- Filter out topics associated to services (RPC).
- Parameterize simulated endpoints.

This release includes the following **Documentation Updates**:

- Update introduction section in README file.

This release includes the following change in the **Continuous Integration** process:

- Update CI to use eProsima CI.

This release includes the following **Dependencies Update**:

| | Repository | Old Version | New Version |
|---|---|---|---|
| Foonathan Memory Vendor | eProsima/foonathan_memory_vendor | v1.3.0 | v1.3.1 |
| Fast CDR | eProsima/Fast-CDR | v1.0.27 | v1.1.0 |
| Fast DDS | eProsima/Fast-DDS | v2.10.1 | v2.11.0 |
| Dev Utils | eProsima/dev-utils | v0.3.0 | v0.4.0 |
| DDS Pipe | eProsima/DDS-Pipe | v0.1.0 | v0.2.0 |

### 3.17.3 Version v0.1.0

This is the first release of *eProsima Fast DDS Spy*.

This release includes several **features** regarding the introspection of DDS data, configuration and user interaction.

This release includes the following **User Interface features**:

- *Application executable arguments*.
- *Interactive application*.
- *One-shot application*.

This release includes the following **Configuration features**:

- Support *YAML configuration file*.
- Support for allow and block topic filters at execution time and in run-time.
- Support *DDS configurations*.
- Support *Advanced configurations*.

This release includes the following **Introspection features**:

- Get data of *Domain Participants*.
- Get data of *DataReaders* and *DataWriters*.
- Get data of *Topics*.
- Print real-time DDS user data in human-readable format.

This release includes the following **Documentation features**:

- This same documentation.

This release includes the following **Installation features**:

- *Dockerfile* to build an image of the application `Fast-DDS-Spy/docker`.

## 3.18 Glossary

**CLI** **Command Line Interface** This is a channel of communication between application and user using the Command Line, so-called Terminal.

**DDS** **Data Distribution Service** protocol. Specification: https://www.omg.org/spec/DDS/.

**YAML** **YAML Ain't Markup Language** human-friendly data serialization language. Specification: https://yaml.org/.

### 3.18.1 DDS nomenclature

**DataReader** DDS element that subscribes to a specific Topic. It belong to one and only one Participant, and it is uniquely identified by a Guid.

See Fast DDS documentation for further information.

**DataWriter** DDS entity that publish data in a specific Topic. It belong to one and only one Participant, and it is uniquely identified by a Guid.

See Fast DDS documentation for further information.

**Discovery Server** Discovery Server Discovery Protocol is a Fast DDS feature that enables a new Discovery mechanism based on a Server that filters and distribute the discovery information. This is highly recommended in networks where multicast is not available (e.g. WAN).

See Fast DDS documentation for further information.

**Domain Id** The Domain Id is a virtual partition for DDS networks. Only DomainParticipants with the same Domain Id would be able to communicate to each other. DomainParticipants in different Domains will not even discover each other.

See Fast DDS documentation for further information.

**DomainParticipant** A DomainParticipant is the entry point of the application to a DDS Domain. Every DomainParticipant is linked to a single domain from its creation, and cannot change such domain. It also acts as a factory for Publisher, Subscriber and Topic.

See Fast DDS documentation for further information.

**Endpoint** DDS element that publish or subscribes in a specific Topic. Endpoint kinds are *DataWriter* or *DataReader*.

**Entity Id** DDS Identifier created by 4 bytes, represented by 4 hexadecimal values separated by `.`. This identifier differentiates each of the entities inside a *DomainParticipant*. e.g. `00.00.01.c1`.

**Guid Prefix** DDS Identifier created by 12 bytes, represented by 12 hexadecimal values separated by `.`. Every entity that belongs to the same *DomainParticipant* has the same Guid Prefix. The EntityId uniquely identifies subentities inside a Participant. e.g. `01.0f.22.ba.3b.47.ab.3c.00.00.00.00`.

**Guid** Global Unique Identifier. It contains a *Guid Prefix* and an *Entity Id*. Identifies uniquely a DDS entity (*DomainParticipant*, *DataWriter* or *DataReader*). e.g. `01.0f.22.ba.3b.47.ab.3c.00.00.00.00|00.00.01.c1`.

**Topic** DDS isolation abstraction to encapsulate subscriptions and publications. Each Topic is uniquely identified by a topic name and a topic type name (name of the data type it transmits).

See Fast DDS documentation for further information.

# INDEX