rostopic

Package Links
Code API
Tutorials
Troubleshooting
FAQ
Changelog
Change List
Reviews

Dependencies (5)
Used by (19)
Jenkins jobs (10)

Package Summary

Released | Continuous Integration: 1371 / 1372 | Documented

rostopic contains the rostopic command-line tool for displaying debug information about ROS Topics, including publishers, subscribers, publishing rate, and ROS Messages. It also contains an experimental Python library for getting information about and interacting with topics dynamically. This library is for internal-use only as the code API may change, though it does provide examples of how to implement dynamic subscription and publication behaviors in ROS.

- Maintainer status: maintained
- Maintainer: Dirk Thomas <dthomas AT osrfoundation DOT org>
- Author: Ken Conley
- License: BSD
- Source: git https://github.com/ros/ros_comm.git (branch: melodic-devel)

Contents

1. rostopic command-line tool

The rostopic command-line tool displays information about ROS topics. Currently, it can display a list of active topics, the publishers and subscribers of a specific topic, the publishing rate of a topic, the bandwidth of a topic, and messages published to a topic. The display of messages is configurable to output in a plotting-friendly format.

rostopic, like several other ROS tools, uses YAML-syntax at the command line for representing the contents of a message. For information on how to use this YAML syntax for commands like rostopic pub, please see the YAML command line guide.

This is the current list of supported commands:

```
rostopic bw     display bandwidth used by topic
rostopic delay display delay for topic which has header
rostopic echo    print messages to screen
rostopic find    find topics by type
rostopic hz     display publishing rate of topic
rostopic info    print information about active topic
rostopic list    print information about active topics
rostopic pub     publish data to topic
rostopic type    print topic type
```

These are described in greater detail in the following sections.

1.1 rostopic bw

bw <topic-name>

Display the bandwidth used by a topic.

```
$ rostopic bw /topic_name
```

NOTE: the bandwidth reported is the received bandwidth. If there are network connectivity issues, or if rostopic cannot keep up with the publisher, the reported number may be lower than the actual bandwidth. rostopic is implemented in Python, which cannot maintain as high throughput as roscpp-based nodes.

1.2 rostopic delay

delay <topic-name>

Display the delay for topic which has header.

```
$ rostopic delay /topic_name
```

1.3 rostopic echo

echo <topic-name>

Display messages published to a topic.

```
$ rostopic echo /topic_name
```

--offset

Display time in messages as offset from current time (e.g. to calculate lag/latency).

```
$ rostopic echo --offset /topic_name
```

--filter

Display messages that match a specified Python expression.

```
$ rostopic echo --filter "m.data=='foo'"  /topic_name
```

The Python expression can use any Python builtins plus the variable m (the message). For example, to filter based on the frame_id of the first transform in a tf/tfMessage:

```
$ rostopic echo --filter "m.transforms[0].child_frame_id == 'my_frame'" /tf
```

-c

Clear the screen after each message is published. Cannot be used with -p.

```
$ rostopic echo -c /topic_name
```

-b

Display messages in a bag file:

```
$ rostopic echo -b log_file.bag /topic_name
```

-p

Display messages in a matlab/octave-friendly plotting format. Cannot be used with -c.

```
$ rostopic echo -p /topic_name
```

-w NUM_WIDTH **New in Indigo**

Print all numeric values with a fixed width.

--nostr, --noarr

Exclude string and array fields from the plotting output.

```
$ rostopic echo -p --nostr --noarr /topic_name
```

-n COUNT **New in C Turtle**

Echo COUNT messages and exit.

echo <topic-name/field>

Display specific fields in a message.

```
$ rostopic echo /my_topic/field_name
```

1.4 rostopic find

find <msg-type>

Find topics by type.

```
$ rostopic find std_msg/String
```

1.5 rostopic hz

hz <topic-name>

Display the publishing rate of a topic. The rate reported is by default the average rate over the entire time rostopic has been running.

```
$ rostopic hz /topic_name
```

NOTE: To get a temporally local estimate of the rate, use the -w option to specify the window size for the average.

-w WINDOW_SIZE

Report rate using a window size (number of samples) for a temporally local estimate of the rate.

--filter FILTER_EXPR **New in C Turtle**

Only report rate for messages that match the Python FILTER_EXPR. WARNING: this option has a large performance hit and shouldn't be used for high-rate topics.

1.6 rostopic info (ROS 0.10)

info <topic-name>

Print information about topic.

```
$ rostopic info clock
```

1.7 rostopic list

list

Display a list of current topics.

```
$ rostopic list
```

list <namespace>

(ROS 0.11) List topics in the specified namespace. In previous versions, this is equivalent to the rostopic info command.

```
$ rostopic list /namespace
```

-b

List topics in a bag file.

-p

List only publishers.

-s

List only subscribers.

-v

Verbose mode.

```
$ rostopic list -v
```

--host **New in Diamondback**

Group list by hostname.

1.8 rostopic pub

pub <topic-name> <topic-type> [data...]

Publish data to a topic.

```
$ rostopic pub /topic_name std_msgs/String hello
```

There are three ways to specify the message fields:

- Command-line arguments. Defaults to *latch mode*. Example usage:

```
$ rostopic pub my_topic std_msgs/String "hello there"
```

- Piped input. Defaults to *rate mode* (10hz). Example usage:

```
$ rostopic echo chatter | rostopic pub bar std_msgs/String
```

- YAML data file. Defaults to *rate mode* (10hz). Example usage:

```
$ rostopic echo chatter > chatter.bagy
Collect messages, then Ctrl-C
$ rostopic pub -f chatter.bagy bar std_msgs/String
```

There are three modes that rostopic can publish in:

Latching mode

rostopic will publish a message to /topic_name and keep it *latched* -- any new subscribers that come online after you start rostopic will hear this message. You can stop this at any time by pressing ctrl-C.

Once mode

If you don't want to have to stop rostopic with ctrl-C, you can publish in *once mode*. rostopic will keep the message latched for 3 seconds, then quit.

Rate mode.

In rate mode, rostopic will publish your message at a specific rate. For example, -r 10 will publish at 10hz. For file and piped input, this defaults to 10hz.

Options:

-l, --latch **New in Diamondback**

Enable latch mode. Latching mode is the *default* when using command-line arguments.

-r RATE

Enable *rate mode*. Rate mode is the *default* (10hz) when using piped or file input.

-1, --once

Enable *once mode*.

-f FILE **New in Diamondback**

Read message fields from YAML file. YAML syntax is equivalent to output of rostopic echo. Messages are separated using YAML document separator ---. To use only the first message in a file, use the --latch option.

Data types are be interpreted using YAML-syntax, e.g. 1 is an integer, 1.0 is a float, and foo is a string.

For example:

```
$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist  '{linear:  {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

or on Windows **New in Melodic**

```
$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist  "{linear:  {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0, z: 0.0}}"
```

Publish a geometry_msgs/Twist message with a rate of 10Hz. **For more a description of the YAML format and some tips for using it on the command-line with ROS, please see YAML command line.**

1.9 rostopic type

type <topic-name>

Display topic type of a topic.

```
$ rostopic type /topic_name
```

This is useful for piping to other commands, like rosmsg, e.g.

```
$ rostopic type /topic_name | rosmsg show
```

2. Roadmap

rostopic is a stable command-line tool within the ROS core toolchain. The underlying code may undergo refactoring for easier library use, but the external API is expected to be fairly stable.

One area in which rostopic is expected to see development is with the output format of rostopic echo and input format of rostopic pub. The planned feature is to make both compatible with YAML syntax, which will enable

- YAML-based message data logging
- YAML-based message data publishing

This feature is currently not scheduled.