

# ROS 2 Cheats Sheet

## Command Line Interface

All ROS 2 CLI tools start with the prefix 'ros2' followed by a verb, a sub-verb and (possibly) positional/optional arguments.

For any tool, the documentation is accessible with,

```
$ ros2 verb --help
```

and similarly for sub-verb documentation,

```
$ ros2 verb sub_verb -h
```

Similarly, auto-completion is available for all (sub-)verbs and most positional/optional arguments. E.g.,

```
$ ros2 verb [tab][tab]
```

Some of the examples below rely on:

[ROS 2 demos package](#).

**action** Allows to manually send a goal and displays debugging information about actions.

Sub-commands:

```
info      Output information about an action.
list      Output a list of action names.
send_goal Send an action goal.
show      Output the action definition.
```

Examples:

```
$ ros2 action info /fibonacci
$ ros2 action list
$ ros2 action send_goal /fibonacci \
  action_tutorials/action/Fibonacci "order: 5"
$ ros2 action show action_tutorials/action/Fibonacci
```

**bag** Allows to record/play topics to/from a rosbag.

Sub-commands:

```
info      Output information of a bag.
play      Play a bag.
record     Record a bag.
```

Examples:

```
$ ros2 info <bag-name>
$ ros2 play <bag-name>
$ ros2 record -a
```

**component** Various component related sub-commands.

Sub-commands:

```
list      Output a list of running containers and components.
load      Load a component into a container node.
standalone Run a component into its own standalone container node.
types     Output a list of components registered in the ament index.
unload    Unload a component from a container node.
```

**daemon** Various daemon related sub-commands.

Sub-commands:

```
start     Start the daemon if it isn't running.
status    Output the status of the daemon.
stop      Stop the daemon if it is running
```

**extension\_points** List extension points.

**extensions** List extensions.

**launch** Allows to run a launch file in an arbitrary package without to cd there first.

Usage:

```
$ ros2 launch <package> <launch-file>
```

Example:

```
$ ros2 launch demo_nodes_cpp add_two_ints.launch.py
```

**lifecycle** Various lifecycle related sub-commands.

Sub-commands:

```
get       Get lifecycle state for one or more nodes.
list      Output a list of available transitions.
nodes     Output a list of nodes with lifecycle.
set       Trigger lifecycle state transition.
```

**msg** Displays debugging information about messages.

Sub-commands:

```
list      Output a list of message types.
package   Output a list of message types within a given package.
packages Output a list of packages which contain messages.
show      Output the message definition.
```

Examples:

```
$ ros2 msg list
$ ros2 msg package std_msgs
```

```
$ ros2 msg packages
```

```
$ ros2 msg show geometry_msgs/msg/Pose
```

**multicast** Various multicast related sub-commands.

Sub-commands:

```
receive   Receive a single UDP multicast packet.
send      Send a single UDP multicast packet.
```

**node** Displays debugging information about nodes.

Sub-commands:

```
info      Output information about a node.
list      Output a list of available nodes.
```

Examples:

```
$ ros2 node info /talker
$ ros2 node list
```

**param** Allows to manipulate parameters.

Sub-commands:

```
delete    Delete parameter.
get       Get parameter.
list      Output a list of available parameters.
set       Set parameter
```

Examples:

```
$ ros2 param delete /talker /use_sim_time
$ ros2 param get /talker /use_sim_time
$ ros2 param list
$ ros2 param set /talker /use_sim_time false
```

**pkg** Create a ros2 package or output package(s)-related information.

Sub-commands:

```
create     Create a new ROS2 package.
executables Output a list of package specific executables.
list       Output a list of available packages.
prefix     Output the prefix path of a package.
xml       Output the information contained in the package xml manifest.
```

Examples:

```
$ ros2 pkg executables demo_nodes_cpp
$ ros2 pkg list
$ ros2 pkg prefix std_msgs
$ ros2 pkg xml -t version
```

**run** Allows to run an executable in an arbitrary package

without having to cd there first.

Usage:

```
$ ros2 run <package> <executable>
```

Example:

```
$ ros2 run demo_node_cpp talker
```

---

**security** Various security related sub-commands.

Sub-commands:

<a href="#">create_key</a>	Create key.
<a href="#">create_permission</a>	Create keystore.
<a href="#">generate_artifacts</a>	Create permission.
<a href="#">list_keys</a>	Distribute key.
<a href="#">create_keystore</a>	Generate keys and permission files from a list of identities and policy files.
<a href="#">distribute_key</a>	Generate XML policy file from ROS graph data.
<a href="#">generate_policy</a>	List keys.

Examples (see [sros2 package](#)):

```
$ ros2 security create_key demo_keys /talker
$ ros2 security create_permission demo_keys /talker \
  policies/sample_policy.xml
$ ros2 security generate_artifacts
$ ros2 security create_keystore demo_keys
```

---

**service** Allows to manually call a service and displays debugging information about services.

Sub-commands:

<a href="#">call</a>	Call a service.
<a href="#">find</a>	Output a list of services of a given type.
<a href="#">list</a>	Output a list of service names.
<a href="#">type</a>	Output service's type.

Examples:

```
$ ros2 service call /add_two_ints \
  example_interfaces/AddTwoInts "a: 1, b: 2"
$ ros2 service find rcl_interfaces/srv/ListParameters
$ ros2 service list
$ ros2 service type /talker/describe_parameters
```

---

**srv** Various srv related sub-commands.

Sub-commands:

<a href="#">list</a>	Output a list of available service types.
<a href="#">package</a>	Output a list of available service types within one package.
<a href="#">packages</a>	Output a list of packages which contain services.
<a href="#">show</a>	Output the service definition.

---

**test** Run a ROS2 launch test.

---

**topic** A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Sub-commands:

<a href="#">bw</a>	Display bandwidth used by topic.
<a href="#">delay</a>	Display delay of topic from timestamp in header.
<a href="#">echo</a>	Output messages of a given topic to screen.
<a href="#">find</a>	Find topics of a given type type.
<a href="#">hz</a>	Display publishing rate of topic.
<a href="#">info</a>	Output information about a given topic.
<a href="#">list</a>	Output list of active topics.
<a href="#">pub</a>	Publish data to a topic.
<a href="#">type</a>	Output topic's type.

Examples:

```
$ ros2 topic bw /chatter
$ ros2 topic echo /chatter
$ ros2 topic find rcl_interfaces/msg/Log
$ ros2 topic hz /chatter
$ ros2 topic info /chatter
$ ros2 topic list
$ ros2 topic pub /chatter std_msgs/msg/String \
  'data: Hello ROS 2 world'
$ ros2 topic type /rosout
```