

ROS 2 Cheats Sheet

All ROS 2 CLI tools start with the prefix 'ros2' followed by a verb and (possibly) a sub-verb. For any tool, the documentation is accessible with,

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

and similarly for sub-verb documentation,

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

action Also allows to manually send a goal and displays debugging information about action.

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.

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:

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

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

Examples:

```
$ ros2 topic bw
$ ros2 topic echo
$ ros2 topic hz
```

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

Sub-commands:

<code>call</code>	Call a service.
<code>find</code>	Output a list of services of a given type.
<code>list</code>	Output a list of service names.
<code>type</code>	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:

<code>list</code>	Output a list of available service types.
<code>package</code>	Output a list of available service types within one package.
<code>packages</code>	Output a list of packages which contain services.
<code>show</code>	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: