

# ROS CHEAT SHEET



## ROS Indigo Cheatsheet

### Filesystem Management Tools

|                          |                                                                           |
|--------------------------|---------------------------------------------------------------------------|
| <b>rospack</b>           | A tool for inspecting <a href="#">packages</a> .                          |
| <b>rospack profile</b>   | Fixes path and pluginlib problems.                                        |
| <b>roscd</b>             | Change directory to a package.                                            |
| <b>rospd/rostd</b>       | <a href="#">Pushd</a> equivalent for <a href="#">ROS</a> .                |
| <b>rosls</b>             | Lists package or stack information.                                       |
| <b>rosed</b>             | Open requested ROS file in a text editor.                                 |
| <b>roscp</b>             | Copy a file from one place to another.                                    |
| <b>roscdep</b>           | Installs package system dependencies.                                     |
| <b>rosvtf</b>            | Displays a errors and warnings about a running ROS system or launch file. |
| <b>catkin_create_pkg</b> | Creates a new ROS stack.                                                  |
| <b>wstool</b>            | Manage many repos in workspace.                                           |
| <b>catkin_make</b>       | Builds a ROS catkin workspace.                                            |
| <b>rqt_dep</b>           | Displays package structure and dependencies.                              |

Usage:

```
$ rospack find [package]
$ roscd [package[/subdir]]
$ rospd [package[/subdir] | +N | -N]
$ rosd
$ rosls [package[/subdir]]
$ rosed [package] [file]
$ roscp [package] [file] [destination]
$ roscdep install [package]
$ rosvtf or rosvtf [file]
$ catkin_create_pkg [package_name] [depend1] .. [dependN]
$ wstool [init | set | update]
$ catkin_make
$ rqt_dep [options]
```

### Start-up and Process Launch Tools

#### roscore

The basis [nodes](#) and programs for ROS-based systems. A roscore must be running for ROS nodes to communicate.

Usage:

```
$ roscore
```

#### roslaunch

Runs a ROS package's executable with minimal typing.

Usage:

```
$ roslaunch package_name executable_name
```

Example (runs [turtlesim](#)):

```
$ roslaunch turtlesim turtlesim_node
```

#### roslaunch

Starts a roscore (if needed), [local nodes](#), [remote nodes](#) via SSH, and sets parameter server [parameters](#).

Examples:

```
Launch a file in a package:
$ roslaunch package_name file_name.launch
Launch on a different port:
$ roslaunch -p 1234 package_name file_name.launch
Launch on the local nodes:
$ roslaunch --local package_name file_name.launch
```

## Introspection and Command Tools

### roscnode

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Commands:

|                         |                                  |
|-------------------------|----------------------------------|
| <b>roscnode ping</b>    | Test connectivity to node.       |
| <b>roscnode list</b>    | List active nodes.               |
| <b>roscnode info</b>    | Print information about a node.  |
| <b>roscnode machine</b> | List nodes running on a machine. |
| <b>roscnode kill</b>    | Kill a running node.             |

Examples:

```
Kill all nodes:
$ roscnode kill -a
List nodes on a machine:
$ roscnode machine aqy.local
Ping all nodes:
$ roscnode ping --all
```

### rostopic

A tool for displaying information about ROS [topics](#), including publishers, subscribers, publishing rate, and messages.

Commands:

|                      |                                          |
|----------------------|------------------------------------------|
| <b>rostopic bw</b>   | Display bandwidth used by topic.         |
| <b>rostopic echo</b> | Print messages to screen.                |
| <b>rostopic find</b> | Find topics by type.                     |
| <b>rostopic hz</b>   | Display publishing rate of topic.        |
| <b>rostopic info</b> | Print information about an active topic. |
| <b>rostopic list</b> | List all published topics.               |
| <b>rostopic pub</b>  | Publish data to topic.                   |
| <b>rostopic type</b> | Print topic type.                        |

Examples:

```
Publish hello at 10 Hz:
$ rostopic pub -r 10 /topic_name std_msgs/String hello
Clear the screen after each message is published:
$ rostopic echo -c /topic_name
Display messages that match a given Python expression:
$ rostopic echo --filter "m.data=='foo'" /topic_name
Pipe the output of rostopic to rosmmsg to view the msg type:
$ rostopic type /topic_name | rosmmsg show
```

### rosservice

A tool for listing and querying ROS services.

Commands:

|                        |                                          |
|------------------------|------------------------------------------|
| <b>rosservice list</b> | Print information about active services. |
| <b>rosservice node</b> | Print name of node providing a service.  |
| <b>rosservice call</b> | Call the service with the given args.    |
| <b>rosservice args</b> | List the arguments of a service.         |
| <b>rosservice type</b> | Print the service type.                  |
| <b>rosservice uri</b>  | Print the service ROSRPC uri.            |
| <b>rosservice find</b> | Find services by service type.           |

Examples:

```
Call a service from the command-line:
$ rosservice call /add_two_ints 1 2
Pipe the output of rosservice to rossrv to view the srv type:
$ rosservice type add_two_ints | rossrv show
Display all services of a particular type:
$ rosservice find rospy.tutorials/AddTwoInts
```

### rosparam

A tool for getting and setting ROS [parameters](#) on the parameter server using YAML-encoded files.

Commands:

|                        |                              |
|------------------------|------------------------------|
| <b>rosparam set</b>    | Set a parameter.             |
| <b>rosparam get</b>    | Get a parameter.             |
| <b>rosparam load</b>   | Load parameters from a file. |
| <b>rosparam dump</b>   | Dump parameters to a file.   |
| <b>rosparam delete</b> | Delete a parameter.          |
| <b>rosparam list</b>   | List parameter names.        |

Examples:

```
List all the parameters in a namespace:
$ rosparam list /namespace
Setting a list with one as a string, integer, and float:
$ rosparam set /foo "{'1', 1, 1.0}"
Dump only the parameters in a specific namespace to file:
$ rosparam dump dump.yaml /namespace
```

### rosmmsg/rossrv

Displays Message/Service (msg/srv) data structure definitions.

Commands:

|                         |                                           |
|-------------------------|-------------------------------------------|
| <b>rosmmsg show</b>     | Display the fields in the msg/srv.        |
| <b>rosmmsg list</b>     | Display names of all msg/srv.             |
| <b>rosmmsg md5</b>      | Display the msg/srv md5 sum.              |
| <b>rosmmsg package</b>  | List all the msg/srv in a package.        |
| <b>rosmmsg packages</b> | List all packages containing the msg/srv. |

Examples:

```
Display the Pose msg:
$ rosmmsg show Pose
List the messages in the nav_msgs package:
$ rosmmsg package nav_msgs
List the packages using sensor_msgs/CameraInfo:
$ rosmmsg packages sensor_msgs/CameraInfo
```

## Logging Tools

### rosvbag

A set of tools for recording and playing back of ROS topics.

Commands:

|                           |                                          |
|---------------------------|------------------------------------------|
| <b>rosvbag record</b>     | Record a bag file with specified topics. |
| <b>rosvbag play</b>       | Play content of one or more bag files.   |
| <b>rosvbag compress</b>   | Compress one or more bag files.          |
| <b>rosvbag decompress</b> | Decompress one or more bag files.        |
| <b>rosvbag filter</b>     | Filter the contents of the bag.          |

Examples:

```
Record select topics:
$ rosvbag record topic1 topic2
Replay all messages without waiting:
$ rosvbag play -a demo_log.bag
Replay several bag files at once:
$ rosvbag play demo1.bag demo2.bag
```

### tf\_echo

A tool that prints the information about a particular transformation between a source\_frame and a target\_frame.

Usage:

```
$ rosvrun tf tf_echo <source_frame> <target_frame>
```

Examples:

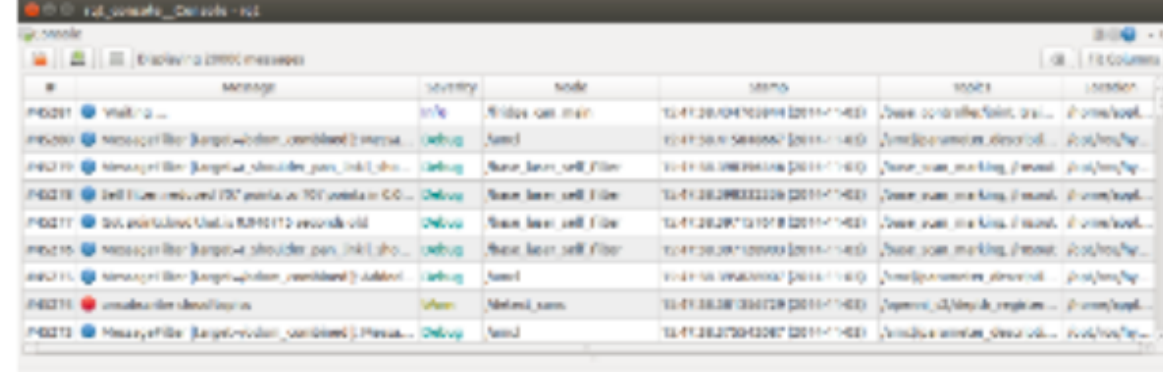
```
To echo the transform between /map and /odom:
$ rosvrun tf tf_echo /map /odom
```

# ROS CHEAT SHEET

## Logging Tools

### rqt\_console

A tool to display and filtering messages published on rosout.

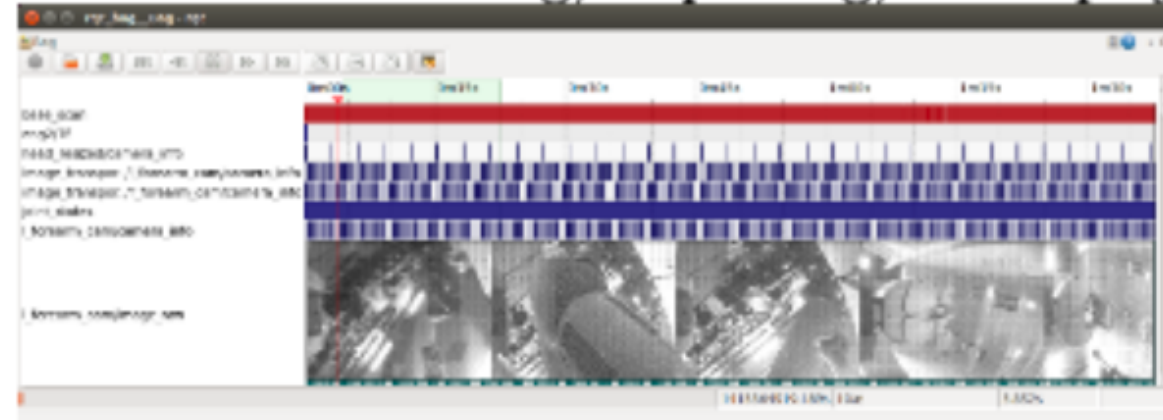


Usage:

```
$ rqt_console
```

### rqt\_bag

A tool for visualizing, inspecting, and replaying bag files.



Usage, viewing:

```
$ rqt_bag bag_file.bag
```

Usage, bagging:

```
$ rqt_bag *press the big red record button.*
```

### rqt\_logger\_level

Change the logger level of ROS nodes. This will increase or decrease the information they log to the screen and rqt\_console.

Usage:

```
viewing $ rqt_logger_level
```

## Introspection & Command Tools

### rqt\_topic

A tool for viewing published topics in real time.

Usage:

```
$ rqt
```

Plugin Menu->Topic->Topic Monitor

### rqt\_msg, rqt\_srv, and rqt\_action

A tool for viewing available msgs, srvs, and actions.

Usage:

```
$ rqt
```

Plugin Menu->Topic->Message Type Browser

Plugin Menu->Service->Service Type Browser

Plugin Menu->Action->Action Type Browser

### rqt\_top

A tool for ROS specific process monitoring.

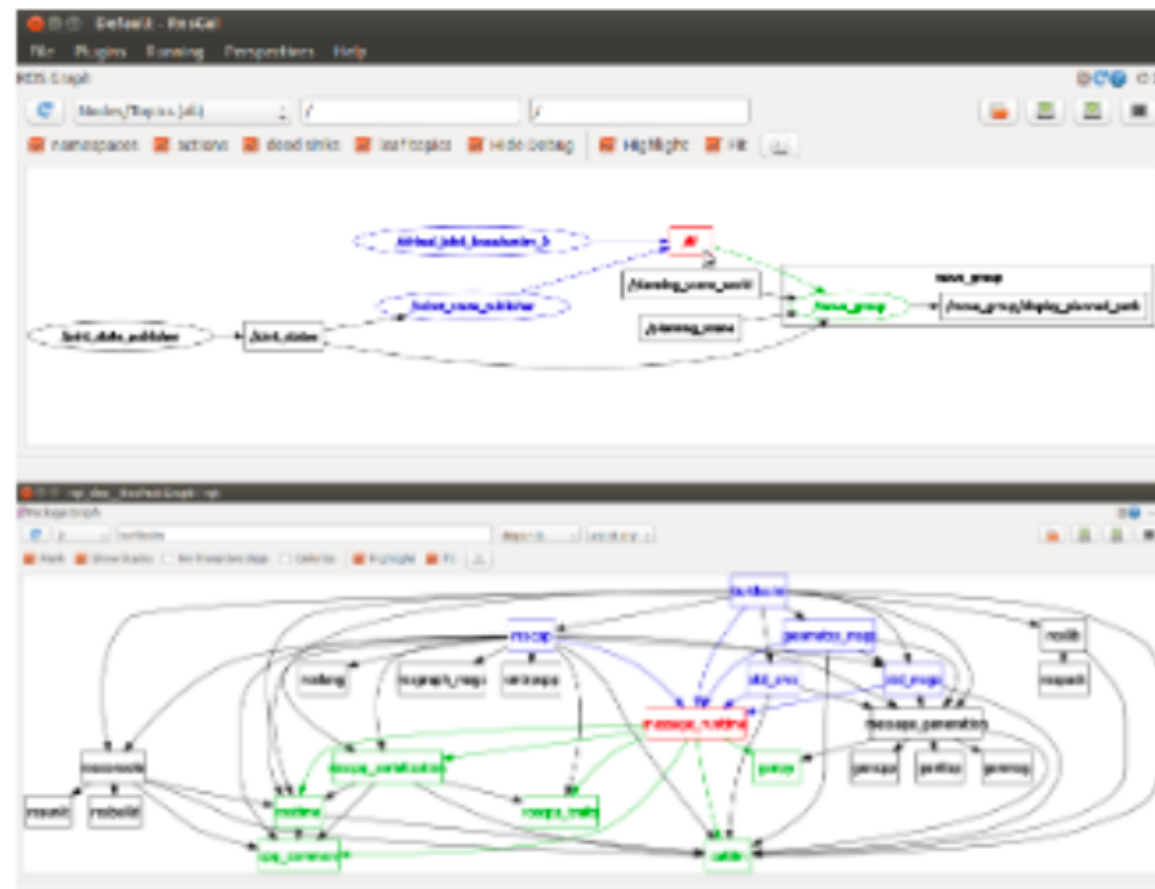
Usage:

```
$ rqt
```

Plugin Menu->Introspection->Process Monitor

### rqt\_graph, and rqt\_dep

Tools for displaying graphs of running ROS nodes with connecting topics and package dependancies respectively.



Usage:

```
$ rqt_graph
```

```
$ rqt_dep
```

### rqt\_publisher, and rqt\_service\_caller

Tools for publishing messages and calling services.

Usage:

```
$ rqt
```

Plugin Menu->Topic->Message Publisher

Plugin Menu->Service->Service Caller

### rqt\_reconfigure

A tool for dynamically reconfiguring ROS parameters.

Usage:

```
$ rqt
```

Plugin Menu->Configuration->Dynamic Reconfigure

## Development Environments

### rqt\_shell, and rqt\_py\_console

Two tools for accessing an xterm shell and python console respectively.

Usage:

```
$ rqt
```

Plugin Menu->Miscellaneous Tools->Shell

Plugin Menu->Miscellaneous Tools->Python Console

## Data Visualization Tools

### view\_frames

A tool for visualizing the full tree of coordinate transforms.

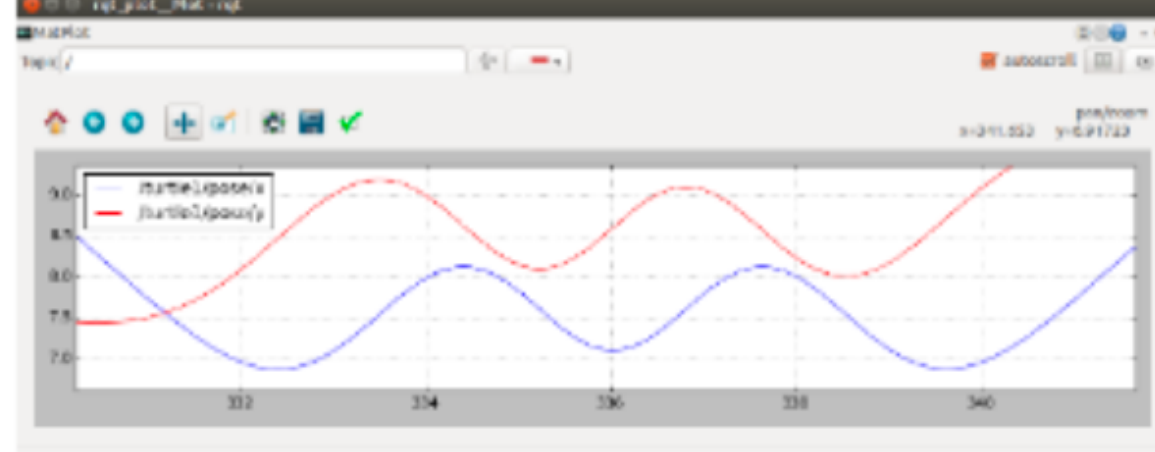
Usage:

```
$ roslaunch tf2_tools view_frames.py
```

```
$ evince frames.pdf
```

### rqt\_plot

A tool for plotting data from ROS topic fields.



Examples:

To graph the data in different plots:

```
$ rqt_plot /topic1/field1 /topic2/field2
```

To graph the data all on the same plot:

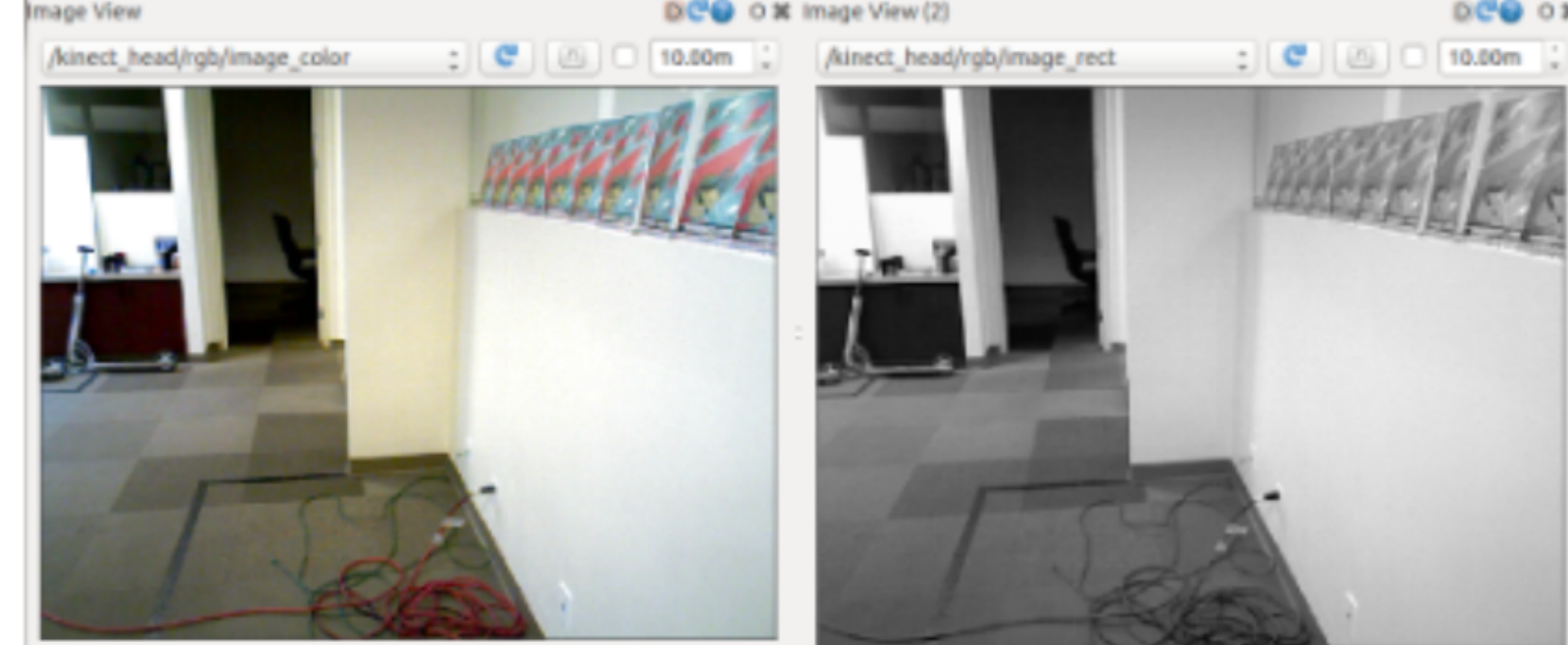
```
$ rqt_plot /topic1/field1,/topic2/field2
```

To graph multiple fields of a message:

```
$ rqt_plot /topic1/field1:field2:field3
```

### rqt\_image\_view

A tool to display image topics.



Usage:

```
$ rqt_image_view
```

## ROS Indigo Catkin Workspaces

### Create a catkin workspace

Setup and use a new catkin workspace from scratch.

Example:

```
$ source /opt/ros/indigo/setup.bash
```

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

### Checkout an existing ROS package

Get a local copy of the code for an existing package and keep it up to date using wstool.

Examples:

```
$ cd ~/catkin_ws/src
```

```
$ wstool init
```

```
$ wstool set tut --git git://github.com/ros/ros_tutorials.git
```

```
$ wstool update
```

### Create a new catkin ROS package

Create a new ROS catkin package in an existing workspace with [catkin create package](#). After using this you will need to edit the `CMakeLists.txt` to detail how you want your package built and add information to your `package.xml`.

Usage:

```
$ catkin_create_pkg <package_name> [depend1] [depend2]
```

Example:

```
$ cd ~/catkin_ws/src
```

```
$ catkin_create_pkg tutorials std_msgs roscpp
```

### Build all packages in a workspace

Use [catkin make](#) to build all the packages in the workspace and then source the setup.bash to add the workspace to the `ROS_PACKAGE_PATH`.

Examples:

```
$ cd ~/catkin_ws
```

```
$ ~/catkin_make
```

```
$ source devel/setup.bash
```

Copyright © 2015 Open Source Robotics Foundation

Copyright © 2010 Willow Garage