

## 5) Writing a basic SF script

[Introducing the ReFlex Hand](#) » [Reflex Documentation](#) » 5) Writing a basic SF script

The next tutorial is [6\) Getting under the hood](#)

In previous tutorials we used the command line to understand how the hand works. Here we'll go through examples of controlling the hand with python. The easiest way to see all the functions available to you is to check `reflex_hand.py`, and `reflex_sf_hand.py`, where the functions are written.

Example code has been written for you. Inside of the reflex package you will find `sf_example_code.py`. You're encouraged to run the code and go through it yourself, commenting out lines or changing lines to see what affect it has on the hand behavior. In this tutorial we will talk about a few of the important moments, but for full understanding you should explore the example script.

### Useful imports

To command the hand a number of ROS messages are used. Note that most of the imports come from `reflex_msgs.msg`, which are the messages defined in the `reflex_msgs` package.

```
from std_srvs.srv import Empty

from reflex_msgs.msg import Command
from reflex_msgs.msg import PoseCommand
from reflex_msgs.msg import VelocityCommand
from reflex_msgs.msg import Hand
```

### Setting up publishers, services, and subscribers

To command your hand you'll likely want some combination of the calls below. In order to automatically calibrate the fingers, you'll need to call the `ServiceProxy` function.

In order to send position/velocity commands to the hand, you'll want to make publishers that publish to the appropriate topic, as shown here.

Finally, if you want to monitor the state of the hand you'll want to make a subscriber to the `/hand_state` topic. That will lets you keep track of the joint positions, motor load, etc.

```
# Services can automatically call hand calibration
calibrate_fingers = rospy.ServiceProxy('/reflex_sf/calibrate_fingers', Empty)

# This collection of publishers can be used to command the hand
command_pub = rospy.Publisher('/reflex_sf/command', Command, queue_size=1)
pos_pub = rospy.Publisher('/reflex_sf/command_position', PoseCommand, queue_size=1)
vel_pub = rospy.Publisher('/reflex_sf/command_velocity', VelocityCommand, queue_size=1)

# Constantly capture the current hand state
rospy.Subscriber('/reflex_sf/hand_state', Hand, hand_state_cb)
```

If you want more information about how to make publishers and subscribers, try checking out the introductory [ROS tutorials on the subject](#).

### Publishing commands

There are many examples of publishing in the example code, but here is a basic illustration. The first thing to do is create a message of the right type, which we do with the `VelocityCommand()` call. The `VelocityCommand` can be populated by setting each of its elements - `f1`, `f2`, `f3`, and `preshape`. If you leave out any elements they will be set to 0.0, which could be desired. Finally, the command is published with `vel_pub.publish()`.

```
vel_pub.publish(VelocityCommand(f1=-5.0, f2=-5.0, f3=-5.0, preshape=0.0))
```

Keep on going to [6\) Getting under the hood](#)

page revision: 4, last edited: 10 Oct 2015, 15:52 (249 days ago)  
[Stop watching site docs.righthandrobotics.com \[2\]](#)

[Edit](#)   [Rate \(0\)](#)   [Tags](#)   [History](#)   [Files](#)   [Print](#)   [Site tools](#)   [+ Options](#)

---

Powered by [Wikidot.com](#)

[Help](#) | [Terms of Service](#) | [Privacy](#) | [Report a bug](#) | [Flag as objectionable](#)

RightHand Robotics Inc (c) 2014