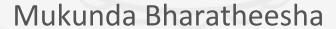
1.5.1

ROS Actions – action messages and goal processing



ROS Actions – generated ROS messages

Messages generated for CounterWithDelay action type.

ROS Actions – generated ROS messages

```
donnie@tudelft:~/ros/hrwros_ws/src/hrwros/hrwros_msgs/msg$ rosmsg show hrwros_msgs/CounterWithDelayGoal
uint32 num_counts

donnie@tudelft:~/ros/hrwros_ws/src/hrwros/hrwros_msgs/msg$ rosmsg show hrwros_msgs/CounterWithDelayResult
string result_message

donnie@tudelft:~/ros/hrwros_ws/src/hrwros/hrwros_msgs/msg$ rosmsg show hrwros_msgs/CounterWithDelayFeedback
uint32 counts_elapsed

donnie@tudelft:~/ros/hrwros_ws/src/hrwros/hrwros_msgs/msg$
```

Messages for use on the application side

ROS Actions – generated ROS messages

Messages generated for CounterWithDelay action type.

ROS Actions – action client

```
hrwros_week1/scripts/counter_with_delay_ac.py
client = actionlib.SimpleActionClient(counter with delay,
CounterWithDelayAction)
client.wait for server()
goal = CounterWithDelayGoal(num_counts = 10)
client.send qoal(qoal)
```

ROS Actions – action message

```
donnie@tudelft:~/ros/hrwros ws/src/hrwros/hrwros msgs/msg$ rosmsg show hrwros msgs/CounterWithDelayAction
hrwros msgs/CounterWithDelayActionGoal action goal
 std msgs/Header header
   uint32 seq
                                      Time stamp and header information
   time stamp
   string frame id
 actionlib msgs/GoalID goal id
                                      Unique goal identifier
   time stamp
   string id
 hrwros_msgs/CounterWithDelayGoal goal Goal message
hrwros msgs/CounterWithDelayActionResult action result
 std msgs/Header header
   uint32 seq
                                      Time stamp and header information
   time stamp
    string frame id
 actionlib msgs/GoalStatus status
   uint8 PENDING=0
   uint8 ACTIVE=1
   uint8 PREEMPTED=2
   uint8 SUCCEEDED=3
   uint8 ABORTED=4
   uint8 REJECTED=5
   uint8 PREEMPTING=6
                                    Different states the goal can be in
   uint8 RECALLING=7
   uint8 RECALLED=8
   uint8 LOST=9
   actionlib msgs/GoalID goal id
     time stamp
     string id
   uint8 status
   string text
 hrwros_msgs/CounterWithDelayResult result
```

ROS Actions – action message

hrwros_msgs/CounterWithDelayActionFeedback action_feedback
 std msgs/Header header

```
uint32 seq
 time stamp
                   Time stamp and header information
 string frame id
actionlib msgs/GoalStatus status
 uint8 PENDING=0
 uint8 ACTIVE=1
 uint8 PREEMPTED=2
 uint8 SUCCEEDED=3
 uint8 ABORTED=4
                   Different states the goal can be in
 uint8 REJECTED=5
 uint8 PREEMPTING=6
 uint8 RECALLING=7
 uint8 RECALLED=8
 uint8 LOST=9
 actionlib msgs/GoalID goal id
    time stamp
    string id
 uint8 status
  string text
hrwros msgs/CounterWithDelayFeedback feedback
 uint32 counts_elapsed Feedback message
```

ROS Actions – action server goal callback

```
hrwros_week1/scripts/counter_with_delay_as.py
self. as = actionlib.SimpleActionServer("counter with delay",
CounterWithDelayAction, self.execute cb=execute cb,
auto start=False)
def execute cb(self, goal):
      #Process any preemption request first.
      for counter idx in range(0, goal.num_counts):
             self. feedback.counts elapsed=counter idx
             rospy.sleep(1.0)
```

ROS Actions – action server topics

```
donnie@tudelft:~/ros/hrwros ws/src/hrwros/hrwros week1/scripts$ rostopic list | grep counter
/counter with delay/cancel
/counter with delay/feedback
/counter with delay/goal
/counter with delay/result
/counter with delay/status
donnie@tudelft:~/ros/hrwros ws/src/hrwros/hrwros weekl/scripts$ rostopic info /counter with delay/feedback
Type: hrwros msgs/CounterWithDelayActionFeedback
Publishers:
 * /counter with delay (http://tudelft:39495/)
Subscribers: None
donnie@tudelft:~/ros/hrwros ws/src/hrwros/hrwros weekl/scripts$ rostopic info /counter with delay/cancel
Type: actionlib msgs/GoalID
Publishers: None
Subscribers:
 * /counter with delay (http://tudelft:39495/)
```

ROS Actions - Remember!!

Attention!

- Simple Action Server/Client: Instance of the generic action server from actionlib ROS package.
- Non-blocking execution of ONE goal at a time.
- New goal to the same action server will pre-empt an active goal.
- Always possible to have your own action server implementation that can process multiple goals.