# ROS 2 Launch

October 2018

William Woodall

open robotics

# Improvements Targeted for ROS 2

- Improve static introspection by expressing intent
- Emphasize use of events to drive behavior
  - i.e. going beyond `required=true` & `respawn=true`
- Verification and deterministic startup
- First-class Python API
  - See `.launch.py` convention later

See design doc pull request:

https://github.com/ros2/design/pull/163

open robotics

# Changes from ROS 1 to ROS 2

- Multiple Nodes per process
  - Need mapping from Nodes to Processes
  - Nodelets in ROS 1 emulated “one node per process” with proxy executables
- Parameters are all node local
  - No truly global parameters
  - What does <param …> or <rosparam …> mean outside of a <node …> tag?

open robotics

# Examples, a Disclaimer

The following examples are using some proposed syntax that is still provisional, but the end result should be close to this.

# Example 1: Talker/Listener

*# package://demo_nodes_cpp/launch/talker_listener.launch.py*

```
$ # can be introspected with:
$ ros2 launch --print-description demo_nodes_cpp talker_listener.launch.py
$ # can be run with:
$ ros2 launch demo_nodes_cpp talker_listener.launch.py
```

```
def generate_launch_description():
    return LaunchDescription([
        ExecuteNodeProcess(
            package='demo_nodes_cpp', node_executable='talker', output='screen'),
        ExecuteNodeProcess(
            package='demo_nodes_cpp', node_executable='listener', output='screen'),
    ])
```

$ lib/demo_nodes_cpp/talker

$ lib/demo_nodes_cpp/listener

open robotics

# Example 1: Talker/Listener

```
# Now with custom node names.

# import statements excluded for brevity.

def generate_launch_description():
    return LaunchDescription([
        ExecuteNodeProcess(
            package='demo_nodes_cpp', node_executable='talker', output='screen',
            node_description=NodeDescription(name='my_talker')),
        ExecuteNodeProcess(
            package='demo_nodes_cpp', node_executable='listener', output='screen',
            node_description=NodeDescription(name='my_listener')),
    ])
```

$ lib/demo_nodes_cpp/talker __node:=my_talker

$ lib/demo_nodes_cpp/listener __node:=my_listener

open robotics

# Example 1: Talker/Listener

```
# Now a process with multiple nodes in it and some custom names.

def generate_launch_description():
    return LaunchDescription([
        ExecuteMultiNodeProcess(
            # manual_composition contains four nodes:
            # talker, listener, and an "add_two_ints" server and client
            package='composition', node_executable='manual_composition', output='screen',
            # only need to describe nodes you want to (re)configure in some way
            node_descriptions={
                'talker': NodeDescription(name='my_talker'),
                'listener': NodeDescription(name='my_listener'),
            }),
    ])
```

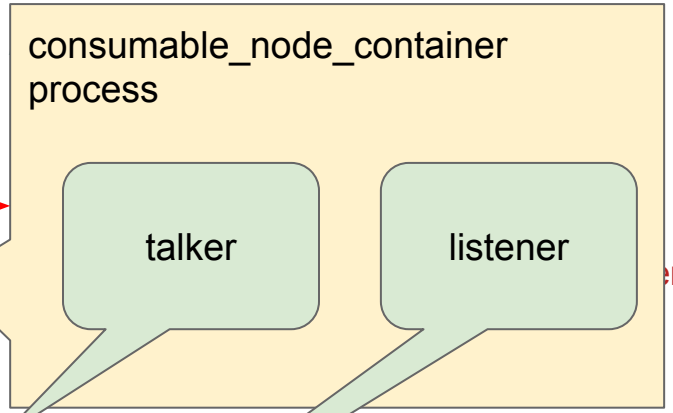
```
$ lib/composition/manual_composition \
    talker:__node:=my_talker \
    listener:__node:=my_listener
```

open robotics

# Example 1: Talker/Listener

```
# Now a process with multiple composable

def generate_launch_description():
    return LaunchDescription([
        ExecuteComposableNodeProcess(
            package='rclcpp_components', n                                  er', output='screen',
            # composable nodes additionally n
            composable_node_descriptions=[
                ComposableNodeDescription(
                  package_name='demo_nodes_cpp', node_plugin_name='talker',
                  name='my_talker'),
                ComposableNodeDescription(
                  package_name='demo_nodes_cpp', node_plugin_name='listener',
                  name='my_listener'),
            ]),
    ])
```




open robotics

# Example 2: Lifecycle Talker/Listener

```
# part 1/2


def generate_launch_description():
    # ...

    # you must describe a lifecycle node with its name
    # even if you do not change the configurations
    talker_node = LifecycleNodeDescription(node_name='talker')

    execute_talker_process_action = ExecuteNodeProcess(
        package='lifecycle', node_executable='lifecycle_talker', output='screen',
        node_description=talker_node)
```

open robotics

# Example 2: Lifecycle Talker/Listener

```
# part 2/2

def generate_launch_description():
    #...
    # When the talker node reaches the 'active' state, log a message and start the listener node.
    register_event_handler_for_talker_reaches_active_state = RegisterEventHandler(
        OnStateTransition(
            target_lifecycle_node=talker_node, goal_state='active',
            entities=[
                LogInfo(
                    msg="node 'talker' reached the 'active' state, launching 'listener'."),
                ExecuteNodeProcess(
                    package='lifecycle', node_executable='lifecycle_listener', output='screen',
                    node_description=LifecycleNodeDescription(node_name='listener')),
            ],
        )
    )
```

open robotics

# Example 2: Lifecycle Talker/Listener

```
# part 2/2

def generate_launch_description():
    #...
    # When the talker node reaches the 'active' state, log a message and start the listener node.
    register_event_handler_for_talker_reaches_active_state = RegisterEventHandler(
        OnStateTransition(
            target_lifecycle_node=talker_node, goal_state='active',
            entities=[
                LogInfo(
                    msg="node 'talker' reached the 'active' state, launching 'listener'."),
                ExecuteNodeProcess(
                    package='lifecycle', node_executable='lifecycle_listener', output='screen',
                    node_description=LifecycleNodeDescription(node_name='listener')),
            ],
        )
    )
```

open robotics

# Example 2: Lifecycle Talker/Listener

```
# part 2/2

def generate_launch_description():
    #...
    # When the talker node reaches the 'active' state, log a message and start the listener node.
    register_event_handler_for_talker_reaches_active_state = RegisterEventHandler(
        OnStateTransition(
            target_lifecycle_node=talker_node, goal_state='active',
            entities=[
                LogInfo(
                    msg="node 'talker' reached the 'active' state, launching 'listener'."),
                ExecuteNodeProcess(
                    package='lifecycle', node_executable='lifecycle_listener', output='screen',
                    node_description=LifecycleNodeDescription(node_name='listener')),
            ],
        )
    )
```

open robotics

# Things I don’t have time to cover

- Substitutions
- Arguments
- Action Conditions
- Including Other Launch Descriptions
- More Actions
- More Events
- More Event Handlers
- Introspection

# Things in Launch you could help with

- Launch descriptions as markup, e.g. XML or YAML
- Verification tools
- Using launch to do testing
- Multi-machine launching

# Where to learn more

- Wiki page:
  - https://github.com/ros2/ros2/wiki/Launch-system
- Architecture document in repository:
  - https://github.com/ros2/launch/blob/master/launch/doc/source/architecture.rst
- Design document pull request:
  - https://github.com/ros2/design/pull/163
- Features targeted for Crystal Release:
  - https://github.com/ros2/launch/issues/101

Questions?

open robotics