RoboLang v1 Standard Library — ROS2 Adapter Edition
==========

Overview

This document defines the RoboLang v1 Standard Library for robot engineers and developers who wish to connect RoboLang tasks directly to ROS2-based robotic systems. The library provides reusable task definitions, along with a Python adapter layer that maps RoboLang primitives to ROS2 actions and services.

Contents

---
1. Mapping: RoboLang → ROS2

RoboLang primitives and corresponding ROS2 entities:

RoboLang primitive	ROS2 entity	Notes
move r to <pose>	FollowJointTrajectory (MoveIt/Arm)	Arm motion
grasp r obj	GripperCommand	Close gripper
place r obj at <pose>	Move + GripperCommand (open)	Placement
inspect r X using sensor	Custom service /inspect_object	Vision system
wait for Ns	Local rclpy sleep	Timing
wait until region_clear region	/region_clear service	Safety check
communicate r to channel with msg	std_msgs/String pub	Communication
sensor_ok name	/sensor_status service	Sensor health

---
2. Python ROS2 Adapter Layer

File: robolang_adapter.py

This adapter implements RoboLang primitives using ROS2.

Example (excerpt):

```
from rclpy.node import Node
from rclpy.action import ActionClient
from std_msgs.msg import String
from control_msgs.action import FollowJointTrajectory, GripperCommand
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint

class RobotAdapter(Node):
    def __init__(self, robot_name: str):
        super().__init__(f"{robot_name}_adapter")
        self.move_client = ActionClient(self, FollowJointTrajectory,
"/arm_controller/follow_joint_trajectory")
        self.gripper_client = ActionClient(self, GripperCommand,
"/gripper_controller/gripper_action")
```

```
        self.comm_publisher = self.create_publisher(String, "/robot_comm", 10)

    def move_to_pose(self, joint_names, joint_positions, duration=2.0):
        goal = FollowJointTrajectory.Goal()
        traj = JointTrajectory()
        traj.joint_names = joint_names
        point = JointTrajectoryPoint()
        point.positions = joint_positions
        point.time_from_start.sec = int(duration)
        traj.points.append(point)
        goal.trajectory = traj
        self.move_client.send_goal_async(goal)

    def set_gripper(self, position=1.0, max_effort=50.0):
        goal = GripperCommand.Goal()
        goal.command.position = position
        goal.command.max_effort = max_effort
        self.gripper_client.send_goal_async(goal)

    def communicate(self, channel, message):
        msg = String()
        msg.data = f"{self.get_name()}:{message}"
        self.comm_publisher.publish(msg)
```

---
3. Example Executors for Common Tasks
---

pick_and_place

RoboLang definition:

```
task pick_and_place(robot r, object obj, location src, location dst, region cell) {
    plan {
        move r to src;
        grasp r obj;
        move r to dst;
        place r obj at dst;
    }
}
```

Python executor:

```
def exec_pick_and_place(adapter, joint_names, src, dst):
    adapter.move_to_pose(joint_names, src)
    adapter.set_gripper(1.0)
    adapter.move_to_pose(joint_names, dst)
    adapter.set_gripper(0.0)
```

inspect_object

```
task inspect_object(robot r, object obj, location cam_pos, region cell, scalar dwell_s) {
    plan {
        move r to cam_pos;
        inspect r obj using "camera_top";
        wait for dwell_s;
    }
}
```

Executor:

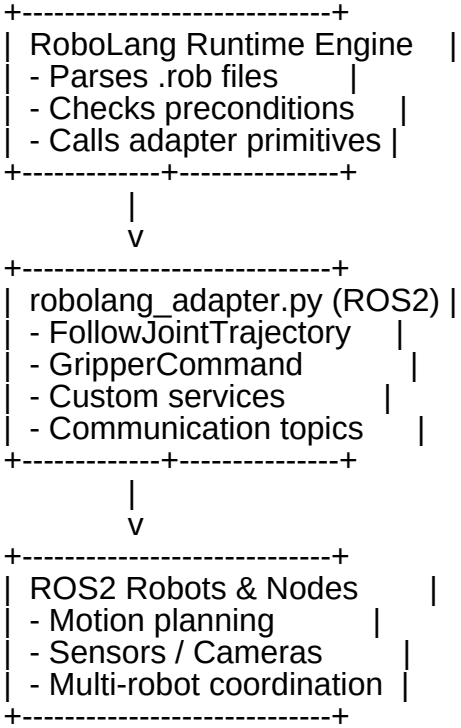
```
def exec_inspect_object(adapter, joint_names, cam_pos, obj_id, dwell_s):
    adapter.move_to_pose(joint_names, cam_pos)
    adapter.inspect(obj_id, "camera_top")
    adapter.wait_for(dwell_s)
```

handover_giver / receiver

Giver:
move to pickup → grasp → move to handoff → communicate HANDOFF_READY
Receiver:
wait until sensor_ok "handoff_signal" → move → grasp

---
4. Integration Notes and Architecture

Typical system layout:



To use this adapter:
1. Include robolang_adapter.py in your ROS2 workspace.
2. Import RobotAdapter and instantiate it per robot.
3. Call methods matching RoboLang primitives.
4. Combine with your RoboLang interpreter or LLM pipeline.

End of RoboLang v1 Standard Library — ROS2 Adapter Edition
