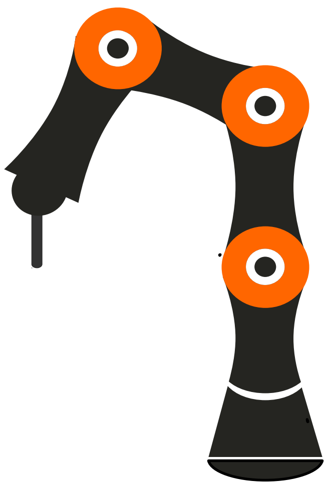
ROS-integrated API for the KUKA LBR iiwa collaborative robot



Saeid Mokaram, Jonathan M. Aitken, James Law

March 2017

Intruduction

Sheffield robotics presents an Application Programming Interface (API) for the KUKA Intelligent Industrial Work Assistant (iiwa) Lightweight Robot (LBR). We have developed this API to support our experimental research work, and is now supporting development of new industrial processes. This API builds upon the safety embedded within the KUKA iiwa to allow close working and interaction with operators. It brings the functionality into the Robot Operating System (ROS), which provides a distributed development environment allowing multiple new modalities of devices to interface easily. This instruction sheet presents ...

API Architecture

The developed API is designed to be simple and interface to ROS to provide an easy platform for development. The API architecture focuses on the breaking out the functionality that would normally be available within KUKA Sunrise controller run on the Smartpad.

The architecture can be viewed to extend the capability of the KUKA LBR iiwa, using the generic structure shown in Figure 1. The API exposes an interface to operation on a network of machines. This allows different sensing methods and extra computing resources to be easily deployed and exploited in operations of the KUKA LBR iiwa.

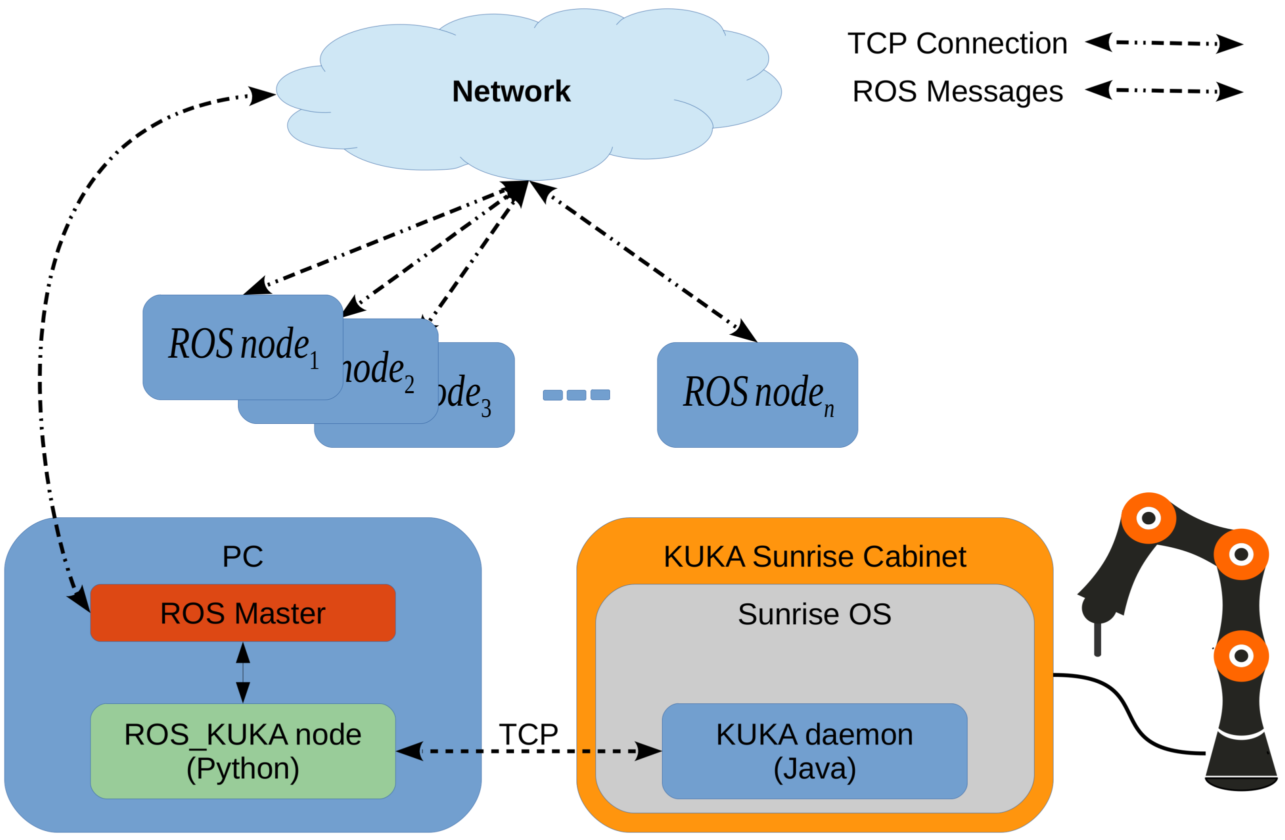


Figure 1. Components of the KUKA ROS interface architecture.

The KUKA daemon also handles some low-level, but generic and critical controlling tasks, such as collision detection. Handling such safety-critical motion controls locally on the Sunrise system is necessary as it provides maximum reliability by using code developed to strict safety standards. This ensures the underlying operation of the robot is still safe, and the presented API forms a wrapper which utilizes these underlying safety-critical protocols.

A ROS-KUKA node is also implemented in the Python scripting language, which is a ROS node and plays an intermediate interface between the KUKA daemon and ROS master. It subscribes to the commands coming from other controlling ROS nodes and passes them to the KUKA server. Similarly it receives sensory and status information from the KUKA server and publishes them under their specific Topics.

Running this ROS-KUKA node externally on a separate computer, rather than on the KUKA Sunrise OS, means that no modification is required of the KUKA Sunrise Cabinet. Therefore the safety protocol provided by KUKA is preserved, and functionality is added rather than altered. Instead of installing a third party software such as ROS on the Sunrise system, the KUKA iiwa ROS interface can be set up with just a standard Sunrise-based application.

The topics available for use within ROS available through the API are shown in Table 1. The KUKA command topic is then used to send instructions through the API to the KUKA LBR iiwa. A list of commands and descriptions is shown in Table 2.

Table 1. Topics available through KUKA iiwa ROS interface. Update on a frequency of 10Hz

Topic Name and Description	Description	Example
JointPosition [A1, A2, A3, A4, A5, A6, A7] time	Joint position (in degrees), reading time-stamp	JointPosition [0.0, 0.17, 0.0, 1.92, 0.0, 0.35, 0.0] 1459253274.1
ToolPosition [X, Y, Z, A, B, C] time	Tool/end-effector position (in cartesian space), reading time-stamp	ToolPosition [433.59711426170867, 0.028881929589094864, 601.4449734558293, 3.1414002368275726, 1.0471367465304213, 3.141453681799645] 1459253274.11
ToolForce [X, Y, Z] time	External force on tool/end effector in different directions, reading time-stamp	ToolForce [13.485958070668463, 0.3785658886199012, 5.964988607372689] 1459253274.11’
ToolTorque [A, B, C] time	External torque on tool/end-effector in different directions, reading time-stamp	ToolTorque [13.485958070668463, 0.3785658886199012, 5.964988607372689] 1459253274.11’
JointAcceleration Float time	Joint acceleration value, reading time-stamp	JointAcceleration 0.4 1459253274.11’
JointVelocity Float time	Joint velocity value, reading time-stamp	JointVelocity 1.0 1459253274.11’
JointJerk Float time	Joint Jerk value, reading time-stamp	JointJerk 1.0 1459253274.11’
isCompliance Boolean time	Robot compliance status, reading time-stamp	isCompliance off 1459253274.11’
isReadyToMove Boolean time	Robot motion status; True if the robot can move or if the robot performed all the motion in its queue, reading time-stamp	isReadyToMove true 1459253274.11’
isCollision Boolean time	True if a collision has detected, reading time-stamp	isCollision false 1459253274.11’

isMastered Boolean time	True if is mastered, reading time-stamp	isMastered true 1459253274.11’
isJointOutOfRange Boolean time	True if any joint is out of its range, reading time-stamp	isJointOutOfRange false 1459253274.11’

Table 2. Topics available through KUKA iiwa ROS interface.

Topic Name and Description	Description	Example
setJointAcceleration F	Setting/changing the joint acceleration value	’setJointAcceleration 0.4’
setJointVelocity F	Setting/changing the joint velocity value	’setJointVelocity 1.0’
setJointJerk F	Setting/changing the joint jerk value	’setJointJerk 1.0’
setPosition A1 A2 A3 A4 A5 A6 A7 ptp/lin	Moving the robot arm based on joint position. Point-to-point (ptp) or linear (lin) motion can be selected. Angular values (in degrees) of type float can be replaced in A1-7. In case any axis doesn’t need to be moved, a - can be used instead of a value. The example assigns new positions for each axis except A2 which doesn’t move.	’setPosition 0 21 0 -100 0 60 0 ptp’
setPositionXYZABC X Y Z A B C ptp/lin	Moving the robot end-effector in the robot cartesian space. Point-to-point (ptp) or linear (lin) motion can be selected. This moves the robot end effector to a particular location [x,y,z] orientation [a,b,c] (values in float). In case any parameter doesn’t need to be changed, a - can be used instead of a value.	’setPositionXYZABC 700 0 290 - 180 0 -180 lin’
MoveXYZABC X Y Z A B C	Moving the robot end-effector in the robot cartesian space with point-to-point (ptp) motion only. This moves the robot end effector in certain direction [x,y,z] and/or orientation [a,b,c] for the given values (in mm and degrees).	MoveXYZABC 10 20 0 30 0 0
MoveCirc X1 Y1 Z1 A1 B1 C1 X2 Y2 Z2 A2 B2 C2 BlendingOri	Moving the robot end-effector in a arch/circular motion from its current position passing from a first one ([x1 y1 z1 a1 b1 c1]) to a second position ([x2 y2 z2 a2 b2 c2]) with a given blending value.	MoveCirc 700 0 290 -180 0 -180 710 0 300 -180 0 -180 0.1
setCompliance X Y Z Z B C	Activates the robot Compliance mode with particular stiffness in each x, y, z, a, b, c. The given example activates the Compliance with a very low stiffness in x and y cartesian plain only.	’setCompliance 10 10 5000 300 300 300’
resetCompliance	Deactivates the robot Compliance mode.	resetCompliance
resetCollision	Resets a Collision if any collision was detected.	resetCollision
forceStop	Stops the robot and removes all the robot motion queue waiting to be executed.	forceStop
setWorkspace xmin ymin zmin xmax ymax zmax	Defining a cubic workspace boundaries.	SetWorkspace 100 -300 0 600 300 500

Using the API

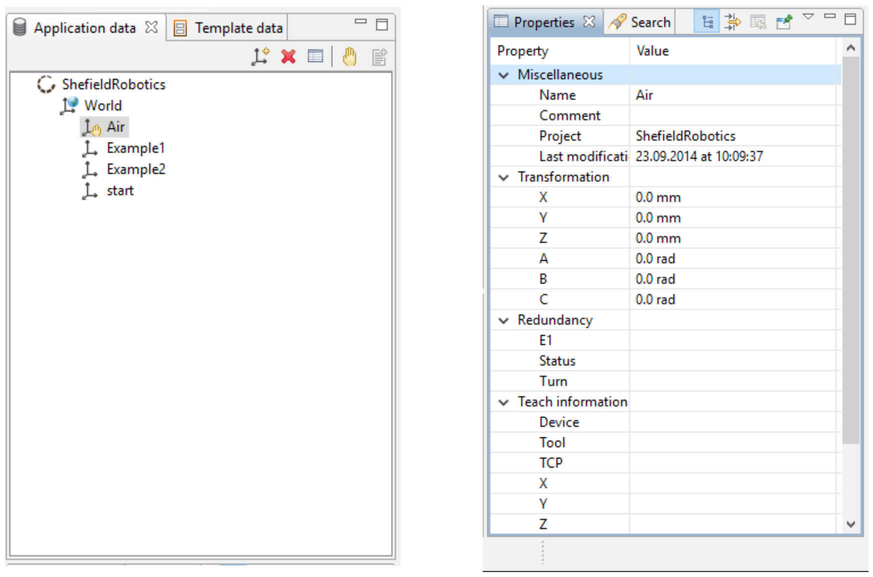
Installing ROS:

To use the API, the first step is to install the ROS. Detailed instructions are provided in: http://wiki.ros.org/ROS/Installation

Installing the API:

As described earlier, the API has two main parts 1) the KUKA daemon (java) and 2) the ROS-KUKA node (Python). The KUKA daemon can be installed on the KUKA Sunrise Cabinet as a standard KUKA application:

1) Import the "ROS_API" folder in to your KUKA Sunrise project as a New Java Package.
2) You need to creating a new frame with the name "Air" (see Figure 2):
   a) Select the project in the Package Explorer.
   b) Right-click on the "World" frame in the Application data view and select Insert new frame from the context menu. The new frame is created and inserted in the frame tree as a child element of the parent frame.
   c) The system automatically generates a frame name. Change the name in the Properties view to "Air".
   d) Make sure all the values for Transformation (x,y,z,a,b,c) is 0.



- ShefieldRobotics
  - World
    - Air
    - Example1
    - Example2
    - start

Property	Value
Miscellaneous	
Name	Air
Comment	
Project	ShefieldRobotics
Last modificati	23.09.2014 at 10:09:37
Transformation	
X	0.0 mm
Y	0.0 mm
Z	0.0 mm
A	0.0 rad
B	0.0 rad
C	0.0 rad
Redundancy	
E1	
Status	
Turn	
Teach information	
Device	
Tool	
TCP	
X	
Y	
Z	

Figure 2. Creating a new frame with the name "Air". Left: Application data – frames. Right: Frame properties by category.

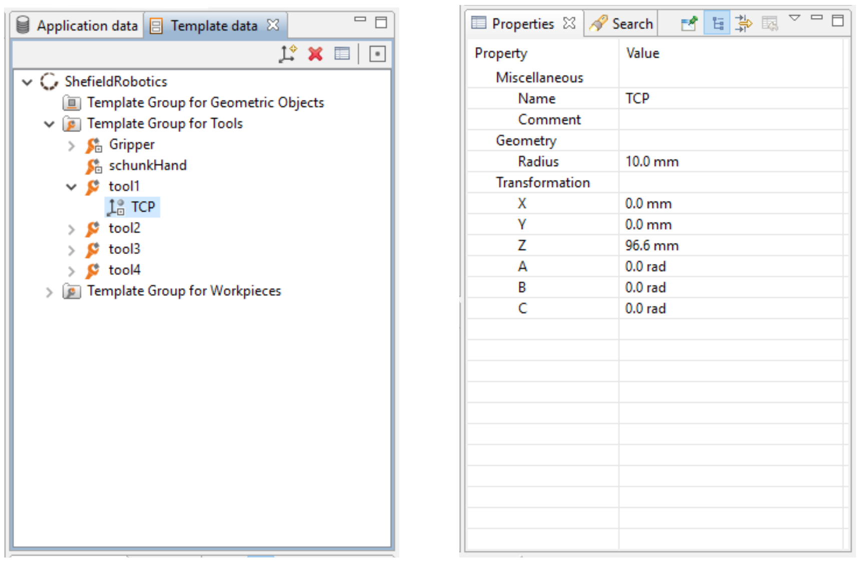
3) Create a default tool (see Figure 3):

   a) Select the project in the Package Explorer.

   b) In the Template data view, open the list of object templates.

   c) Right-click on the Template Group for Tools and select Insert new tool with the name "tool1".

   d) In the Properties view, enter the transformation data of the frame with respect to its parent frame (X, Y, Z: offset of the frame along the axes of the parent frame) (A, B, C: orientation of the frame with reference to the parent frame)



Property	Value
Miscellaneous	
Name	TCP
Comment	
Geometry	
Radius	10.0 mm
Transformation	
X	0.0 mm
Y	0.0 mm
Z	96.6 mm
A	0.0 rad
B	0.0 rad
C	0.0 rad

Figure 3. Creating a default tool.

4) The KUKA daemon connects to the ROS-KUKA node server by default at IP address of 172.31.1.50 and Port:1234. If it is required, change the this address in the “API_ROS_KUKA_V30032017.java’’ at line: skt = new Socket("172.31.1.50", 1234);

5) All the motion commands are executed trough a function called “MoveSafe”. This function applies an external force control on all the motion. A motion terminates if an external force detected. The sensitivity of the MoveSafe can be modified by changing the “sensCLS” value in the “defineSensitivity()” function. You may need to change the default value depending on your task.

6) Synchronize your project with the robot controller.

To install the KUKA node:

1) Copy the “kuka_controller” folder in you ROS workspace.

2) Check the IP address of ROS server.

3) Modify "server 172.31.1.50 port 1234" in the conf.txt file if it is required.

Running a demo in Python:

In a terminal run:

- roscore
- rosrun kuka_controller server_V30032017.py
- On the robot control pad run: “API_ROS_KUKA_V30032017”

Now the API is up and running. The final step is to run a ROS node to communicate with the API and control the robot. Some basic examples are provided in Python. For instance, to start you can run:

- rosrun kuka_controller demo5_ReadData.py

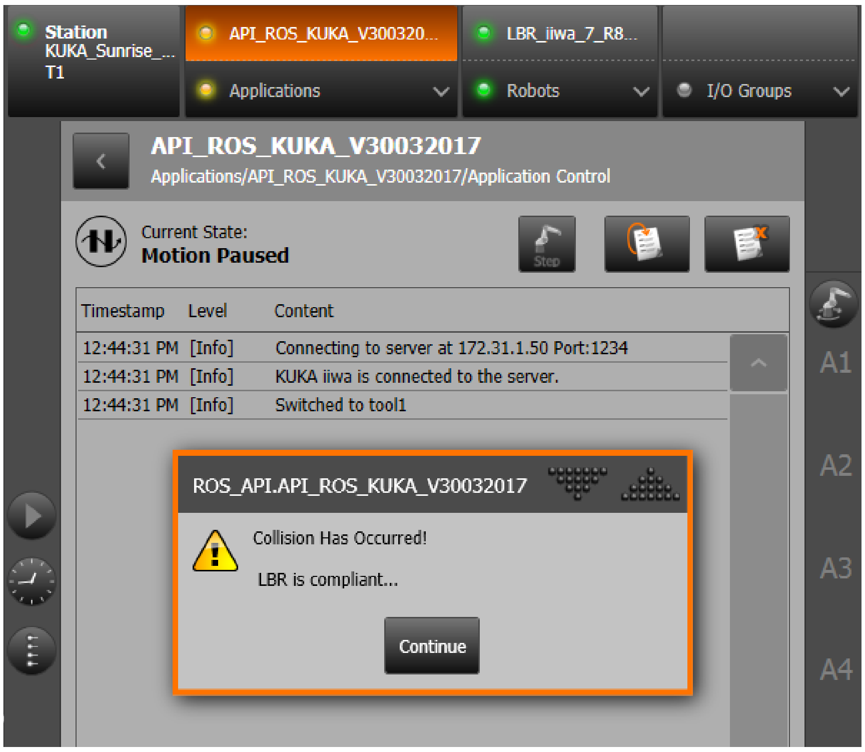
In this demo the robot does not move. This demo receives the robot data like joints positions etc. and print them out. Other demos provide a basic code for performing different motions.

Notifications and errors:

If an undefined text command is sent to the robot, an error appears on the robot control pad such as:

"Unacceptable 'setPosition' command!"

If a collision happens while the robot is moving, the robot stops and goes into a soft compliant mode and a warning box (see Figure 4) appears on the robot control pad:



API_ROS_KUKA_V30032017
Applications/API_ROS_KUKA_V30032017/Application Control

Current State:
Motion Paused

```
Timestamp    Level    Content
12:44:31 PM  [Info]   Connecting to server at 172.31.1.50 Port:1234
12:44:31 PM  [Info]   KUKA iiwa is connected to the server.
12:44:31 PM  [Info]   Switched to tool1
```

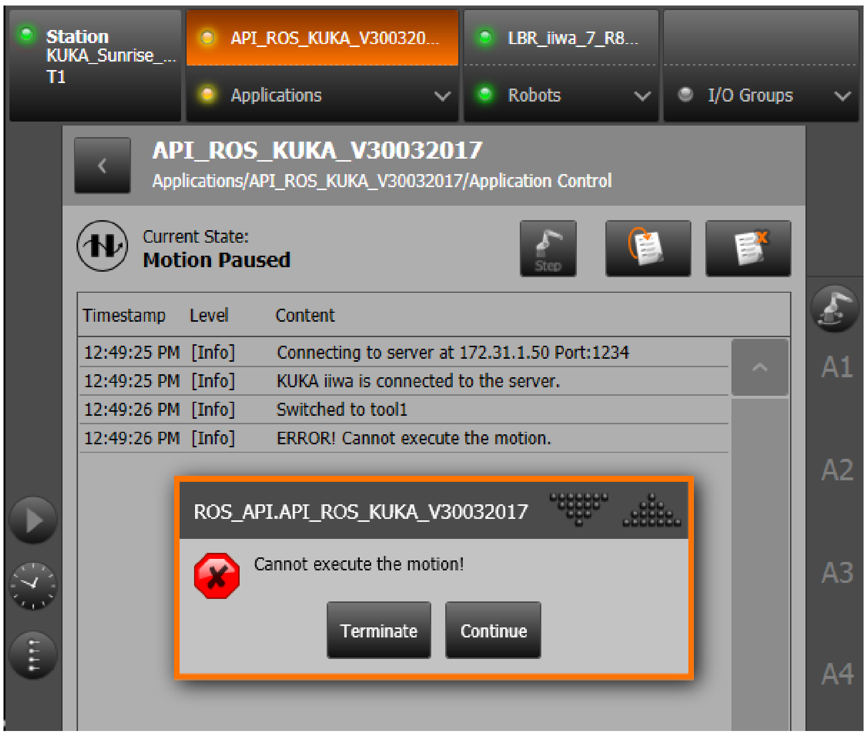
ROS_API.API_ROS_KUKA_V30032017

Collision Has Occurred!

LBR is compliant...

Figure 4. An example of a collision warning.

If a motion command (e.g MoveXYZABC 2000 20 0 30 0 0) cannot be executed because of a reason like trying to move the robot tool to a position wich is out of the robot reach, an error box (see Figure 5) appears on the robot control pad. You can choose to terminate the entire process by pressing “Terminate” or just ignoring that specific motion command and executing the following commands by selecting “Continue”.



```
Timestamp    Level    Content
12:49:25 PM  [Info]   Connecting to server at 172.31.1.50 Port:1234
12:49:25 PM  [Info]   KUKA iiwa is connected to the server.
12:49:26 PM  [Info]   Switched to tool1
12:49:26 PM  [Info]   ERROR! Cannot execute the motion.
```

ROS_API.API_ROS_KUKA_V30032017

Cannot execute the motion!

Figure 5. A motion error example.