XML-RPC communication

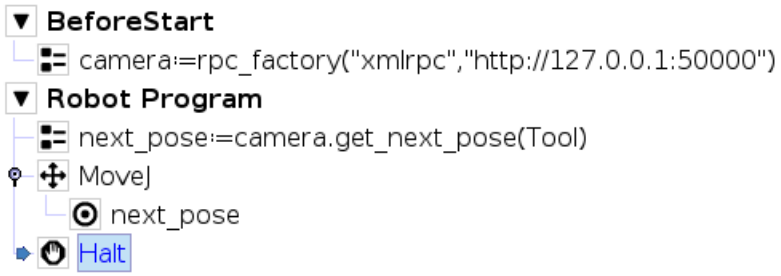
XML-RPC is a Remote Procedure Call method that uses XML to transfer data between programs over sockets. With it, the UR controller can call methods/functions (with parameters) on a remote program/server and get back structured data. The UR controller handles writing the XML-RPC client messages, i.e. it handles all the details of translating between UR types and XML on the wire.

For many programming languages free XML-RPC servers are available, among others for: Python, Java, C++ and C. In here we provide a simple example where we want to make the robot move using a pose retrieved from a fake remote camera. The remote "camera" program, providing the next target pose, is exemplified in both Python and C++.

Please also consult the UR script manual for more information on the XML-RPC.

PolyScope program

PolyScope program xmlrpc_example.urp:



In the BeforeStart sequence we make the connection to the external program and give it the name "camera". In the Robot Program body we ask for the next pose by calling the function get_next_pose on the camera, with the current Tool pose as an argument. The get_next_pose is implemented by the remote program and requires in this example one pose as an input argument. The UR controller will automatically make all of the necessary conversions to communicate over XML-RPC with the remote program (including conversion of the pose). During the execution of the get_next_pose function the PolyScope program waits. When the get_next_pose returns with a pose, the MoveJ moves the robot to the next_pose.

Some attention points:

- It is important that connections to the remote program are built-up in the BeforeStart sequence, since it takes a relatively long time to build up the connection compared to performing the function call. When the "camera" is not used it causes no overhead.
- XML-RPC function calls are transfered to remote programs. Depending on the computing power & network speed the reaction time for the get_next_pose might vary.
- When the program stops, the XML-RPC connection will be automatically cleaned up.
- It is recommended to use other ports than occupied ports and frequently used port such as 8080 to avoid conflicts.

Python

The example Python “camera” program receives a pose and sends a fixed pose back. The PolyScope program is in charge of moving the actual robot.

Python 2.7 & 3 code, xmlrpc_camera.py:

```
import sys
import urlib
is_py2 = sys.version[0] == '2'
if is_py2:
  from SimpleXMLRPCServer import SimpleXMLRPCServer
else:
  from xmlrpc.server import SimpleXMLRPCServer

def get_next_pose(p):
  assert type(p) is dict
  pose = urlib.poseToList(p)
  print("Received pose: " + str(pose))
  pose = [-0.18, -0.61, 0.23, 0, 3.12, 0.04];
  return urlib.listToPose(pose);

server = SimpleXMLRPCServer(("", 50000), allow_none=True)
server.RequestHandlerClass.protocol_version = "HTTP/1.1"
print("Listening on port 50000...")

server.register_function(get_next_pose, "get_next_pose")

server.serve_forever()
```

Running the example

This example uses a Linux PC (for example Ubuntu 14.04).

0) Download the xmlrpc_camera.py and urlib.py

1) Open a terminal

2) Navigate to the downloaded file

3) Type in the following command and press enter: python xmlrpc_camera.py

4) Now the server runs and can be contacted

5) Download the xmlrpc_example.urp and transfer it to the robot

6) Replace the IP-address in the “http://127.0.0.1:50000” string with the one from the Linux PC, for example: “http://192.168.1.10:50000”.

7) Press play to run the PolyScope program

If everything works well the robot moves to the fixed pose. The Python “camera” server will give an output that will be similar to:

```
127.0.0.1 - - [26/Jan/2015 15:50:21] "POST / HTTP/1.1" 200 -
Received pose:  [-0.18, -0.61, 0.23, 0, 3.12, 0.04]
```

Note, the “Received pose” will be equal to the robot's Tool pose when it was started.

C++

This example shows the same functionality as for Python. For C++ the free library XMLRPC-C [http://xmlrpc-c.sourceforge.net/] can be used and is available in most standard Linux distributions by default. Furthermore, its website also features a lot of useful examples and hints to make the C++ programming easier. To install on Ubuntu 14.04 type:

```
sudo apt-get update

sudo apt-get install libxmlrpc-c++8 g++
```

Running the example

0) Download the xmlrpc_cpp.tar.gz

1) Open a terminal and unpack the tar.gz with: tar xf xmlrpc_cpp.tar.gz

2) Compile the code using g++: make

3) Start the executable with: ./xmlrpc_camera

4) Now the server runs and can be contacted

5) Download the xmlrpc_example.urp and transfer it to the robot

6) Remove the “http://” and replace the IP-address in the “http://127.0.0.1:50000” string with the one from the Linux PC, for example: “129.168.1.10:50000”.

7) Press play to run the PolyScope program

If everything works well the robot moves to the fixed pose. The C++ “camera” server will give an output that will be similar to:

```
Received pose: p[-0.18, -0.610001, 0.368829, 0, 3.12, 0.04]
```

Note, the “Received pose” will be equal to the robot's Tool pose when it was started.