23/10/2021

IO 6: WEBLAB TUTORIAL OF TECHNICAL DESIGN AND IMPLEMENTATION – SERVER IMPLEMENTATION (LABVIEW)

Co-funded by
the European Union

1. INTRODUCTION

A remote lab interface developed with EjsS can easily communicate with the lab hardware/software using the Remote Interoperability Protocol (RIP). For this, two things are needed: EjsS' RIP element (see the **Client implementation** manual) and a RIP server implementation.

This manual describes how to use the RIP server implemented in LabVIEW. There is another one for the Python implementation of the RIP server (see the **Server implementation (Python)** manual). The LabVIEW implementation addressed in this document is extremely useful when the lab equipment is controlled, of course, with LabVIEW Virtual Instruments (VIs). Chapter 2 of this manual explains how to configure the RIP server and use it to publish your VIs as web services that can be consumed by EjsS applications with the RIP element.

But first, you need to know where to get the LabVIEW implementation of the RIP Server. The software is available at https://github.com/UNEDLabs/rip-labview-server. For the most advanced users (those who want to do some development), a document presenting the specification of the Remote Interoperability Protocol can be found here: https://github.com/UNEDLabs/rip-spec.

.

2. USING THE LABVIEW IMPLEMENTATION OF THE RIP SERVER

The LabVIEW implementation of the RIP server is distributed as a stand-alone project. Figure 1 shows the files and directories contained in the project. The main file is the *RIPWebSerice.lvproj* and the place where your VIs should be placed is inside the *Private* directory.

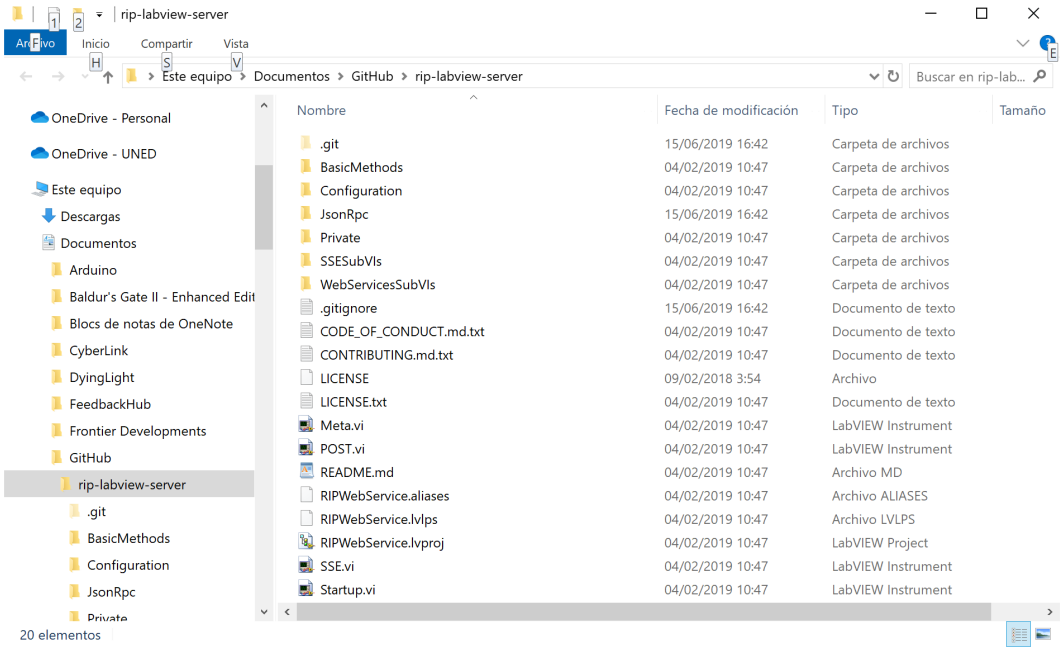

Figure 1. Files belonging to the LabVIEW project for the RIP server

The following sections will guide you on how to: 1) get your LabVIEW programs to work with RIP (2.1), 2) start running the RIP server (2.2) and 3) test your LabVIEW programs actually work through RIP.

Before that, however, keep in mind you will need to have LabVIEW 2015 or later to make this RIP server work.

2.1. Making your VIs work with the RIP server

To get your LabVIEW program published as a RIP web service, the first step is to copy your VI files to the Private directory. Two VIs are distributed with the RIP server by default for testing purposes: *JiLTestDoNotClose.vi* and *JiLTestOK.vi*. You can either place your VI files directly in here or, even better, create a subdirectory, named as your experiment, for example, and place them inside.

The second step is to provide the RIP server with the required information for it to actually know where the VIs and what meta data information is associated to this control program. For this, go to the *Configuration* directory and double-click on the *Global_Configurations.vi* file to open it in LabVIEW. Figure 2 shows and example on how it should look. Here, let's ignore the elements at the top part of the VI and focus on the *Experiments* list. This contains all experiments defined in the current RIP server. As said before, two examples are provided for testing purposes. Accordingly, both are included in the list of experiments. Each experiment is defined here through the fields listed in Table 1. While it is recommended to include information in all of them, the only required ones are *Name* and *Path*. Also, the *Cameras* element may not be of any importance at all if you are using ENLARGE to access your remote labs. In these cases, it is ENLARGE

who serves the information about the cameras used in the remote experience, not RIP, and it is in the ENLARGE system (the myFrontier or myFrontier+ platforms) where cameras must be defined, not here.

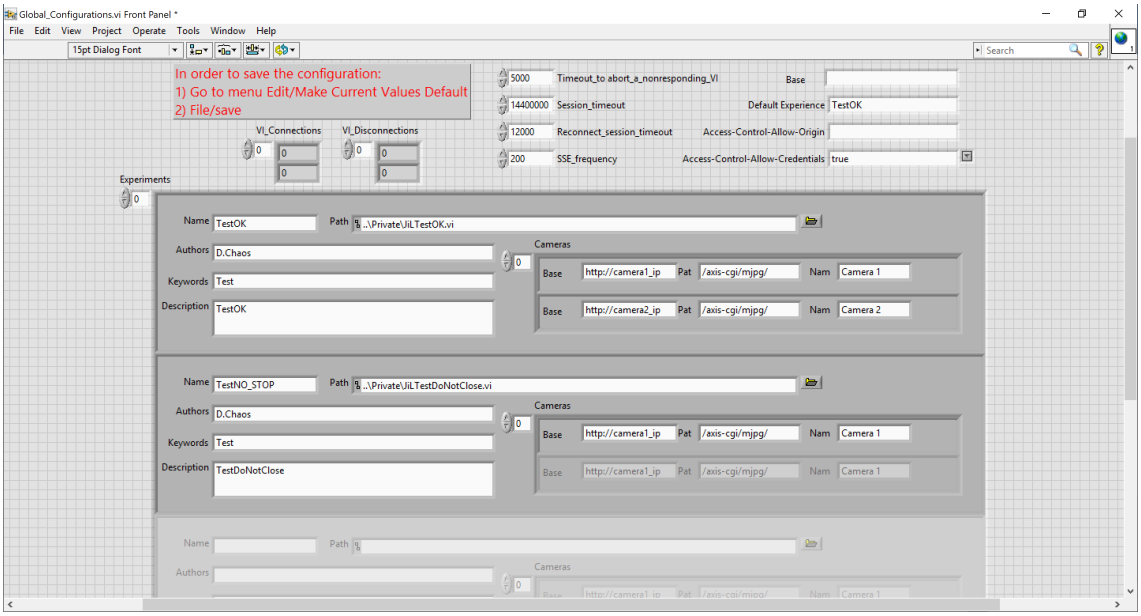

Figure 2. Configuration file to get your LabVIEW programs working with the RIP server

Table 1. Parameters that define a remote experience or experiment in RIP

Field	Description
Name	The ID or name given to the remote experience. This will be used to let the client tell the RIP server which VI it wants to communicate with.
Path	The path (either absolute or relative, but relative is recommended) to the main VI of your LabVIEW control program application.
Authors	The authors of the LabVIEW program.
Keywords	Some keywords related to the remote experience, lab setup, etc.
Description	A description of the remote experience, lab setup, etc.
Cameras	A list of URLs and paths to the cameras used by the remote experience. Not important if you are using ENLARGE.

If you add a new experiment, remember, as the Front Panel marks with red texts, to go to menu *Edit* and click on the option *Make Current Values Default* before saving and closing the file.

2.2. Running the RIP server

Running the RIP server is extremely simple and works just like any other web service project in LabVIEW. To run the RIP server, first double-click on the *RIPWebSerice.lvproj* and a window similar to the one in Figure 3 will open. The right image in the figure shows how to start the web service to get the RIP server started. The process consists on right-clicking on *RIP* in the tree of elements of the LabVIEW project and then selecting *Application Web Server* > *Publish*. By detault, the web server is deployed in port 8080. Note that if you click on *Start* instead of in *Application Web Server* > *Publish*, the server will run in debug mode and will listen to port 8001 instead. Moreover, if you deploy the RIP server using the *Publish* option, the system will automatically deploy it again every time the computer reinitializes.

A new window will then open showing the deployment progress (Figure 4). Once the process has finished, click on the *Close* button to close the window. You can also mark the *Close on successful completion* to avoid having to manually close the window every time you deploy the service.

Now, you are ready to start using your VIs through RIP.

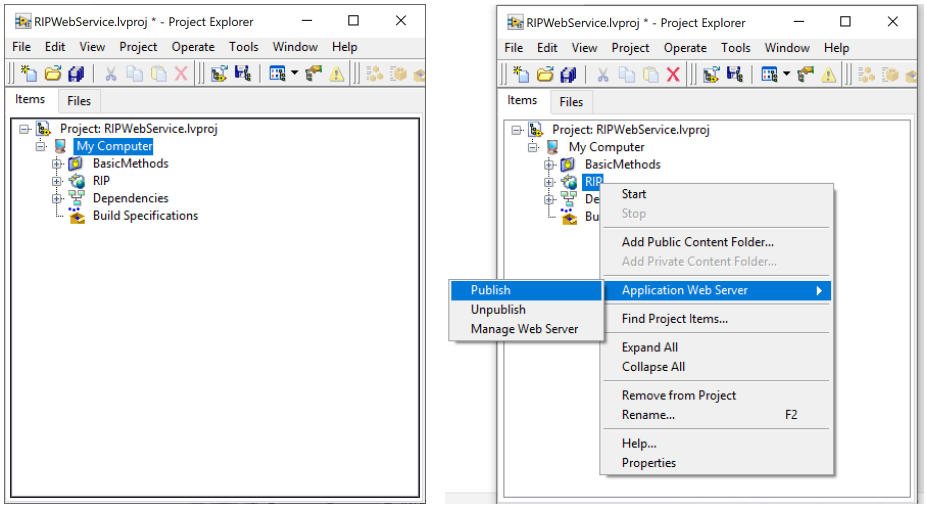

Figure 3. RIP server web service LabVIEW project

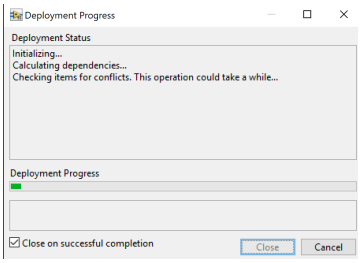

Figure 4. Deployment progress window

2.3. Testing your LabVIEW program is working with the RIP server

Before testing your LabVIEW program, first test if the RIP serer is actually running. For that, open your web browser and enter this URL: http://localhost:8080/RIP. You should see something like Figure 5.

Figure 5. Testing the RIP server

If you get a response from the RIP server in the form of a JSON, then your RIP server is up and running. Not only that, if you already modified the *Global_Configuration.vi* fiile to include information about your remote experiment, you should see information about it in the received JSON structure. Indeed, looking at Figure 5 we can see that the RIP server is sending us information about the two default experiences: *TestOK* and *TestNO_STOP*. If you prefer, you can use a JSON parser to get a better view of the contents. There are online JSON parsers for this.

Now let's see if RIP is able to communicate with one of the default LabVIEW programs; for example, the one associated to the *TestOK* identifier. Type the following in your web browser navigation bar: http://localhost:8080/RIP?expID=TestOK. The web browser should now display something similar to Figure 6.

Figure 6. Testing the RIP server

Again, a JSON parser helps a lot to visualize and understand the information contained in the received JSON. Also, you can read the RIP specification to get a better idea on what information provides each field: https://github.com/UNEDLabs/rip-spec. However, this is only necessary if you intend to contribute to RIP somehow with new developments.

Finally, you can test your own LabVIEW program and RIP-enabled remote experiment by typing http://localhost:8080/RIP?expID=YourProgramID, where *YourProgramID* would be the name or ID you had entered in the *Name* field in the *Global_Configurations.vi* file when you added your experiment.