Resource Control

Universal Robots A/S

Version 1.11.0

Abstract

As of URCap API version 1.7, a resource control framework is introduced for version 3.10.0/5.4.0 of PolyScope. This document will describe how to request exclusive control of a system resource, and how to work with the resource once the user has granted control to the URCap.

Contents

1 Introduction

This document describes the concept of system resources, how to request exclusive control of them, and general dos and don'ts.

2 Purpose of Resource Control

Resources on the robot is defined as an entity that can be controlled. A resource can be unique or a limited number of that resource type can be present on the robot.

In this document, the Tool I/O Interface resource will be the running example. This is a unique resource in the sense that there is only one resource of this kind available on the robot. The Tool I/O Interface is typically used by a device attached to the robot tool, such as a gripper.

The resource control framework will ensure, that the resource can only be controlled by one controller at a time. This controller can be the end user or a URCap requester. While only one URCap or the end user can modify a specific controllable resource, the current resource configuration or state can, at any time, be read by all URCaps (including URCaps which do not control the resource).

Note that in the case of the Tool I/O Interface resource, the URCap in control only controls the configuration of this interface, not the actual values of the I/O signals in the robot tool. Any non-controlling URCap or the end user is free to modify the value of any of these I/Os.

3 How to Request Exclusive Control of a Resource

To request exclusive control of the Tool I/O Interface, a controller for this must be created and registered. This controller will receive a callback when the end user assigns control to the requesting URCap.

The Tool I/O Interface is a resource belonging to the installation section of PolyScope and must therefore be requested using the `ControllableResourceModel` interface accessible from the `InstallationAPI` interface. Once an instance of the `ControllableResourceModel` interface has been obtained, the `requestControl(ToolIOInterfaceController)` method can be invoked passing in the created controller.

When the end user assigns control, the `onControlGranted(ToolIOInterfaceControlEvent)` method is called on the controller. The passed `ToolIOInterfaceControlEvent` parameter will contain the resource instance itself. This can now be used to control all the Tool I/O Interface settings, e.g. the tool voltage.

It is guaranteed that no other URCap or the end user can control the resource, until the `onControlToBeRevoked(ToolIOInterfaceControlEvent)` method is called on the controller. This happens when the end user reassigns control, as well as when a new installation is created or a different installation is loaded and when the robot is shut down. While in the scope of the `onControlToBeRevoked(...)` method, the controller still has exclusive control and can use the time to safely power off the device attached to the robot tool or any similar graceful shutdown procedure.

It is not possible nor necessary to release control of a resource programmatically. This is up to the end user to decide when and if a URCap should no longer be in control.

4 Dos and Don'ts

While the previous section stated that exclusive control was guaranteed, this only applies for PolyScope. It is still possible to use script code to circumvent the exclusive resource control at runtime. This behaviour is no longer allowed. All control and configuration of resources must happen through the resource control framework.

Even if the URCap still has control, it is not allowed to generate script code for configuring the resource. This will be handled by PolyScope using the settings applied by the controlling URCap.

If a URCap does not control a resource, the user assigned control to another URCap (or himself). In that situation, consider if the URCap can be "compatible" with a controlling URCap (or another attached device in case the resource is hardware-based). If that is the case, then the URCap should read the resource configuration to validate that it can, at any time, run with the current settings and report to the end user, if that is not the case (and provide instructions on how to assign the URCap the control).

It is possible to keep a reference to the resource instance passed when the `onControlGranted(...)` method (described in previous section) is called on the controller. This can be useful, if the URCap wants to allow the end user to change some settings of the resource after the control has been granted to the URCap.

This can be a valid approach as long as a check is made to verify that the control has not been reassigned in the meantime. This is done by calling the `hasControl()` method on the resource instance before invoking a setter method (to configure resource settings). This is only applicable when outside the scope of either the controller's `onControlGranted(...)` method or `onControlToBeRevoked(...)` method, since exclusive control is guaranteed when inside the scope of said functions.

5 Code Examples

Listing 1 shows an installation node requesting control of the Tool I/O Interface using a dedicated controller class. This class is also responsible for determining, if help should be shown to the end user, if the URCap is not in control. This help section will guide the end user to resolve the issue by directing him to the Tool I/O installation screen where he can assign the URCap control of the resource.

Finally, the controller also validates whether the URCap can run with the current Tool I/O Interface settings, even if it is not in control (e.g. the end user is manually controlling the settings or the controlling URCap is using a compatible configuration). This will be used by the associated program node to determine its defined-state (not shown).

Notice how the `generateScript(ScriptWriter)` method is empty, since all the script code for configuring the Tool I/O Interface is executed by PolyScope using the settings applied by the controller class.

Listing 1: Requesting control from an installation node

```
1  public class ToolIOControlInstallationNodeContribution implements
       InstallationNodeContribution {
2    private final ToolIOControlInstallationNodeView view;
3    private final ToolIOController toolIOController;
4
5    public ToolIOControlInstallationNodeContribution(InstallationAPIProvider
         apiProvider, ToolIOControlInstallationNodeView view) {
6      this.view = view;
7
8      ControllableResourceModel resourceModel = apiProvider.getInstallationAPI()
           .getControllableResourceModel();
9      toolIOController = new ToolIOController(resourceModel);
10     resourceModel.requestControl(toolIOController);
11   }
12
13   @Override
14   public void openView() {
15     view.showHelp(!toolIOController.hasControl());
16   }
17
18   @Override
19   public void closeView() {
20
21   }
22
23   @Override
24   public void generateScript(ScriptWriter writer) {
25
26   }
27
28   public boolean isSetupCorrect() {
29     return toolIOController.isSetupCorrect();
30   }
31 }
```

Listing 2 shows the Tool I/O Interface controller. It implements the `ToolIOInterfaceController` interface and other methods used by the installation node mentioned previously. When the `onControlGranted(ToolIOInterfaceControlEvent)` method is called, the Tool I/O Interface resource instance is stored in a member variable and the device attached to the robot tool is powered on (by setting the tool voltage to 24V).

When the `onControlToBeRevoked(ToolIOInterfaceControlEvent)` method is called, the URCap is still in control of the resource and uses it to shut down the device (by setting the tool voltage to 0V). The remaining methods implement functionality used by the installation node.

Listing 2: Tool I/O interface controller

```
1  class ToolIOController implements ToolIOInterfaceController {
2
3    private static final OutputVoltage REQUIRED_VOLTAGE = OutputVoltage.
         OUTPUT_VOLTAGE_24V;
4    private static final OutputVoltage SHUTDOWN_VOLTAGE = OutputVoltage.
         OUTPUT_VOLTAGE_0V;
5
6    private final ToolIOInterface toolIOInterface;
7    private ToolIOInterfaceControllable controllableInstance;
8
9    public ToolIOController(ResourceModel resourceModel) {
10     toolIOInterface = resourceModel.getToolIOInterface();
11   }
12
13   @Override
14   public void onControlGranted(ToolIOInterfaceControlEvent event) {
15     controllableInstance = event.getControllableResource();
16     powerOnTool();
17   }
18
19   @Override
20   public void onControlToBeRevoked(ToolIOInterfaceControlEvent event) {
21     shutDownTool();
22   }
23
24   public boolean hasControl() {
25     return controllableInstance != null && controllableInstance.hasControl();
26   }
27
28   private void powerOnTool() {
29     controllableInstance.setOutputVoltage(REQUIRED_VOLTAGE);
30   }
31
32   private void shutDownTool() {
33     controllableInstance.setOutputVoltage(SHUTDOWN_VOLTAGE);
34   }
35
36   public boolean isSetupCorrect() {
37     return toolIOInterface.getOutputVoltage() == REQUIRED_VOLTAGE;
38   }
39 }
```