Capabilities

Universal Robots A/S

Version 1.14.0

Abstract

As of URCap API version 1.7, the concept of capabilities supported by the underlying robot/system is introduced for version 3.10.0/5.4.0 of PolyScope. This document will describe how to work with this capability concept.

Contents

1 Introduction

A capability is a system feature which may or may not be present on a given robot system. The feature can be hardware-based or software-based, such as certain types of program nodes. It can be available on the robot system depending on its generation, specific robot type or purchased options.

2 How to Query for a Capability

When using areas of the URCap API where a given method or interface is referring to a capability, it must first be verified whether the capability is available using the `CapabilityManager` interface. After successful verification, the aforementioned method or interface can be used.

Failing to do so will result in a `CapabilityNotSupportedException` being thrown, if the capability is not present on the robot system.

3 Which API Areas Require Capability Checks

All capabilities are Java enums implementing the `Capability` marker interface. They are located in sub-packages of the `com.ur.urcap.api.domain.system.capability` package.

Listing 1 shows an example of the declaration of such a capability `enum` (representing Tool I/O Interface capabilities) located in the `tooliointerface` sub-package.

Listing 1: Example of a declaration of a `Capability` enum

```
1  public enum ToolIOCapability implements Capability {
2
3    /**
4     * <p>
5     * Capability to configure the analog inputs in the tool to be used as a
          Tool Communication Interface (TCI)
6     * </p>
7     *
8     * Note: This functionality is not available on CB3 robots
9     */
10   COMMUNICATION_INTERFACE_MODE,
11
12   /**
13    * <p>
14    * Capability to configure output mode of the digital outputs in the tool
15    * </p>
16    *
17    * Note: This functionality is not available on CB3 robots
18    */
19   DIGITAL_OUTPUT_MODE
20
21 }
```

Another way is to inspect the Javadoc and annotations of a method or entire interface. This will state whether or not it is necessary to check beforehand, if the capability (that is not guaranteed to be available on all systems) is present on the underlying system.

If a method or interface requires an availability check, the `RequiredCapability` annotation will mention which specific type of `enum` to use with the `hasCapability(Capability)` method in the

CapabilityManager interface. If the annotation is on an interface, then an availability check should be performed before using any of the methods in that interface.

An example of a method annotated with RequiredCapability is shown in Listing 2 (in this case for the Tool Output Mode feature of the Tool I/O Interface). Listing 3 shows an example of an interface with the RequiredCapability annotation.

Listing 2: Example of a method with the RequiredCapability annotation

```
1 public interface ToolIOInterface {
2   ...
3   @RequiredCapability(DIGITAL_OUTPUT_MODE)
4   DigitalOutputModeConfig getDigitalOutputModeConfig();
5   ...
6 }
```

Listing 3: Example of an interface with the RequiredCapability annotation

```
1 @RequiredCapability(DIGITAL_OUTPUT_MODE)
2 public interface DigitalOutputModeConfigFactory {
3   ...
4 }
```

Below in Table 1 is a list of capabilities in the URCap API as of version 1.14.

Capability enum	Interface	Method
ProgramNodeCapability		
SCREWDRIVING	ScrewdrivingNode	all methods
SCREWDRIVING	ProgramNodeFactory	createScrewdrivingNode()
ToolIOCapability		
DIGITAL_OUTPUT_MODE	ToolIOInterface	getDigitalOutputModeConfig()
DIGITAL_OUTPUT_MODE	ToolIOInterfaceControllable	setDigitalOutputModeConfig(...)
DIGITAL_OUTPUT_MODE	ToolIOInterfaceControllable	getDigitalOutputModeConfigFactory()
DIGITAL_OUTPUT_MODE	DigitalOutputModeConfigFactory	all methods
COMMUNICATION_INTERFACE_MODE	CommunicationInterfaceConfig	all methods
COMMUNICATION_INTERFACE_MODE	AnalogInputModeConfigFactory	createCommunicationInterfaceConfig(...)

Table 1: Capabilities in URCap API version 1.14

4 Code Example

Below in Listing 4 is an example of how to check if the Tool Communication Interface (TCI) feature is available on the underlying robot system. If this capability is present, a configuration of this interface is applied.

Note that to configure the TCI feature, it is required to have exclusive control of the Tool I/O Interface (not shown in this example). This is described in the document *Resource Control*.

Listing 4: Checking the availability of a system capability

```
1 ...
2 CapabilityManager capabilityManager = apiProvider.getSystemAPI().
      getCapabilityManager();
```

```
3
4  if (capabilityManager.hasCapability(ToolIOCapability.
       COMMUNICATION_INTERFACE_MODE)) {
5    AnalogInputModeConfigFactory factory =
6      controllableInstance.getAnalogInputModeConfigFactory();
7
8    CommunicationInterfaceConfig config =
9      factory.createCommunicationInterfaceConfig(baudrate, parity, stopbits,
           rxidle, txidle);
10
11   controllableInstance.setAnalogInputModeConfig(config);
12 }
13 ...
```