Modify Execution of Robot Statements

Version: 5 January 2018

You can use a Python Process Handler to modify the execution of statements in a robot program.

# How It Works

By default, a Robot Executor is responsible for executing its program. This involves calling a sequence of statements during a simulation. When a statement is called it is executed by the robot.

A Python Process Handler can be assigned to a Robot Executor to define how a robot executes a called statement. For example, the handler can extend statements, ignore statements, move the robot to other positions, and call other statements without affecting the call sequence of the executor.

## Step 1. Teach Robot

Generally, you would first teach a robot and write its program.



## Step 2. Create Python Process Handler

A Python Process Handler is a behavior used for customizing a robot program. For example, you can define Process type statements and modify the execution of built-in statements.



## Step 3. Assign Handler to Executor

You need to assign the handler to the executor to affect the robot program.



## Step 4. Define script for statement execution

If the handler is assigned to the executor, you need to use the script of that handler to execute statements.

**Important**: Process statements and Path statements with assigned process handlers are executed by their handlers. That is, those statements are not passed on to the statement handler of executor. Do not call nor try to execute them with the statement handler.

Process Handler

Statement Handler

### Case 1. Skip all statements

In this case, the executor executes its program, but its handler will override and skip each statement.



**Note:** This will not skip statements that are not passed on to the handler. For example, the executor would still execute Process statements in its program.

### Case 2. Skip some statements

In this case, the handler will skip some statements but execute others. Generally, a statement has an added property that is used to define a condition for skipping it.



### Case 3. Extend statements by type

In this case, the handler extends a type of statement.



### Case 4. Customize statements by type

In this case, the handler customizes a type of statement. Generally, this is done for Comment statements that contain keywords and other data for executing user-defined statements. For example, a robot program might contain statements written in a different programming language that were imported as comments.



**Note:** Extend or customize? This document considers <<extend>> to mean you are adding functionality to a statement, whereas <<customize>> indicates you are completely modifying a statement to execute a custom set of actions.

### Case 5. Execute a Process statement

In this case, the handler should not be used to execute a Process statement. In all cases, a Process statement should have a separate a handler, which does not need to be associated with the executor. During runtime, the executor will refer to the handler of the Process statement. This is also true for any Path statement that has its own process handler.



### Case 6. Modify a motion statement

In this case, the handler is used to modify the movement of a robot.

Linear and point-to-point (PTP) motion statements contain only one position. To work around this issue, you can assign new targets for the robot to move to either before or after the position of the statement. You can also get configuration warnings, handle reachability and other limit issues, and try to avoid collisions.



### Case 7. Execute a Path statement

A Path statement can contain many positions, which you can edit on the fly.



A safe approach is to assign a process handler to a Path statement. Any Path statement that has an assigned handler will not be passed on to the statement handler of executor. That is, the executor will refer to the handler of the Path statement during runtime.



### Case 8. Call a different statement or routine

In this case, the handler is used to call another statement or routine. This can be a local or remote call. You have the option of waiting until the called statement/routine is finished or immediately continuing the execution of script. You also have the option of clearing the call stack of the executor, so the program would not continue after executing the called routine or statement.



# API

## vcExecutor

### Methods

|  |
| --- |
| callRoutineReturn Type: NoneParameters: vcRoutine routine, [Boolean suspendScript], [Boolean clearCallStack]Executes a given routine.An optional suspendScript argument can be given a False value, thereby Python execution continues immediately.An optional clearCallStack argument can be given a False value, thereby the call stack of executor is not cleared before executing routine. |
| callStatementReturn Type: NoneParameters: vcStatement statement, [Boolean suspendScript]Executes a given statement.An optional suspendScript argument can be given a False value, thereby Python execution continues immediately.**Note:** You can use callStatement() with a process handler to modify the default execution of statements in robot program, for example point-to-point motions, as well as the pre/post execution of statements. |