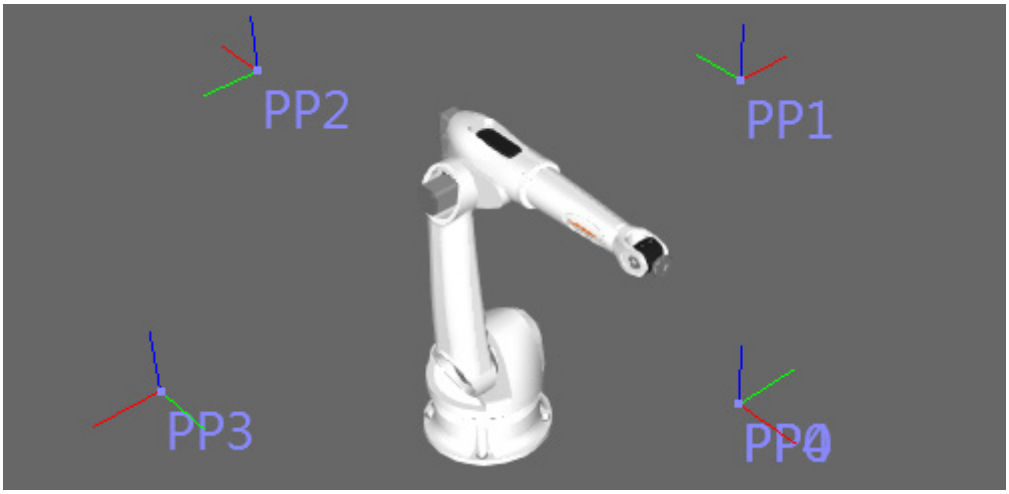
Teach a Process Statement

Visual Components 4.0 | Version: March 27, 2017



A Process statement allows a robot to execute a user-defined process. A process is first defined by the script of a Process Handler, which can be written in .NET or Python API. A robot statement implemented as a Process statement is then used to call the handler of that process.

In this tutorial you learn how to:

- Use a Python Process Handler behavior to define a process.
- Add, remove and modify process positions.
- Create and execute a Process statement in a robot program.

Support
support@visualcomponents.com

Community
community.visualcomponents.net

Define Process

A Process Handler is used to define a process that can be executed by a robot.

1. Add a robot to the 3D world, for example a Generic Articulated Robot available in the Visual Components Web eCatalog.

2. Click the Modeling tab, and then in the Behavior group, click the **Behaviors** arrow, and then under Misc, click **Process Handler**. The script editor will open automatically when you add the behavior.

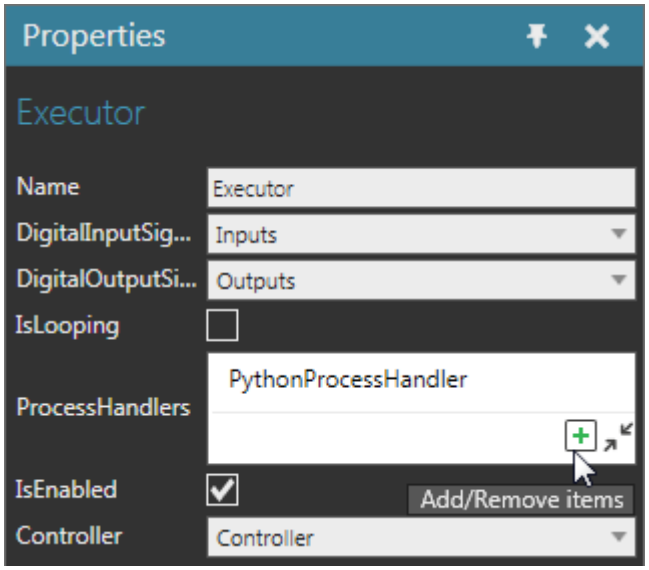
3. In the script editor, add the following code to define process positions and a robot controller will move to each position.

```
from vcPythonProcessHandler import *


def OnStatementAdd(statement):
  for i in range(5):
    pos = statement.createPosition("PP")
    mtx = pos.PositionInWorld
    mtx.translateAbs(1400,0,1000)
    mtx.rotateAbsZ(i*90)
    pos.PositionInWorld = mtx


def OnStatementExecute(executor, statement):
  for pos in statement.Positions:
    executor.Controller.moveTo(pos.PositionInWorld)
```

4. In the Component Graph panel, select the **Robot Executor** behavior of the component, and then in the Properties panel, set ProcessHandlers to **PythonProcessHandler**.



Execute Process

A Process statement can be added to a robot program and used to call an executable process.

1. On the Modeling tab, in the Behavior group, click the **Behaviors** arrow, and then under Misc, click **Python Script**.

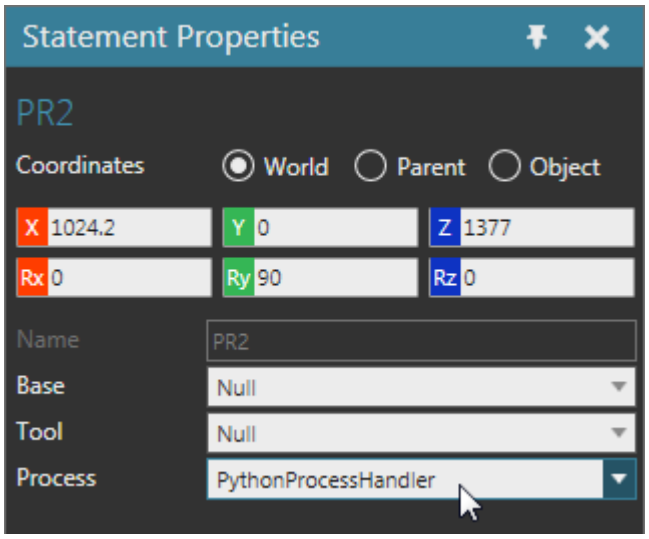
2. In the Python Script editor, add the following code to insert a new Process statement in the main routine of the robot, and then compile the code.

```
from vcScript import *


comp = getComponent()
rx = comp.findBehavioursByType("rRobotExecutor")[0]
rx.Program.MainRoutine.addStatement("Process")
```

3. Click the **Program** tab, and then in the Program Editor panel, select the **process statement** in the main routine.

4. In the Statement Properties panel, set Process to **PythonProcessHandler**, the one you assigned to the robot executor in the previous section. This will add robot positions defined in the process handler script to the statement.



5. Run the simulation, verify the robot moves to each position in the process, and then reset the simulation.

Modify Process

You can modify a process by editing the script of its process handler. For example, you can rename the positions of a process and remove them from the 3D world.

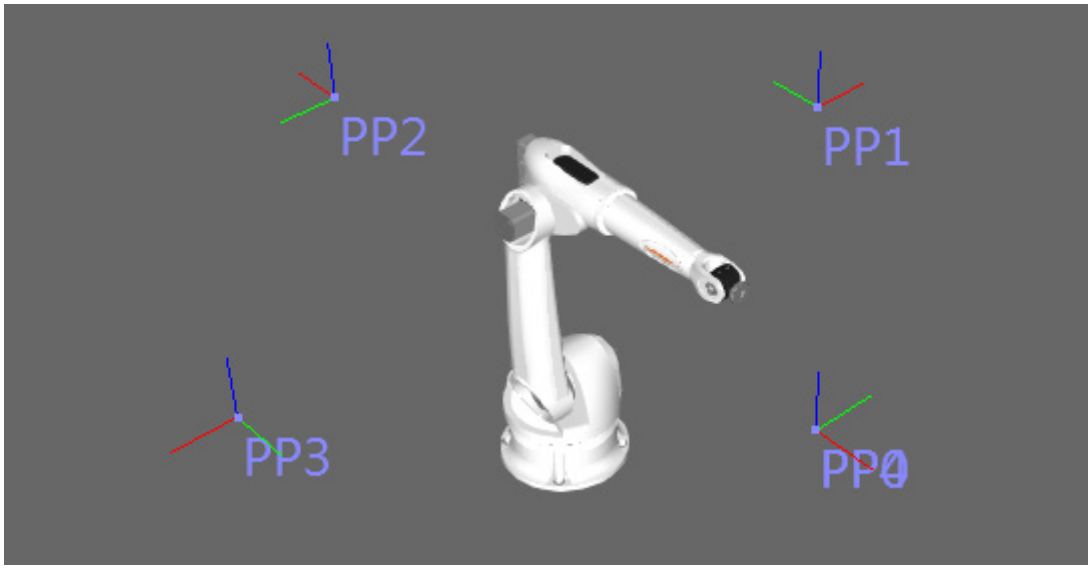
1. Access the PythonProcessHandler script editor, and then modify the OnStatementAdd event to give each position a unique name, and then use the OnStatementRemove event to clear process positions from the 3D word, and then compile the code.

```
def OnStatementAdd(statement):
  for i in range(5):
    pos = statement.createPosition("PP")
    #modify name
    pos.Name += str(i)
    mtx = pos.PositionInWorld
    mtx.translateAbs(1400,0,1000)
    mtx.rotateAbsZ(i*90)
    pos.PositionInWorld = mtx
  getApplication().render()


def OnStatementRemove(statement):
  for pos in statement.Positions:
    statement.deletePosition(pos)
  getApplication().render()
```

NOTE! When adding/removing robot positions, rendering the 3D world is optional.

2. Test your modifications by changing the Process property of the robot statement to null and then to PythonProcessHandler.



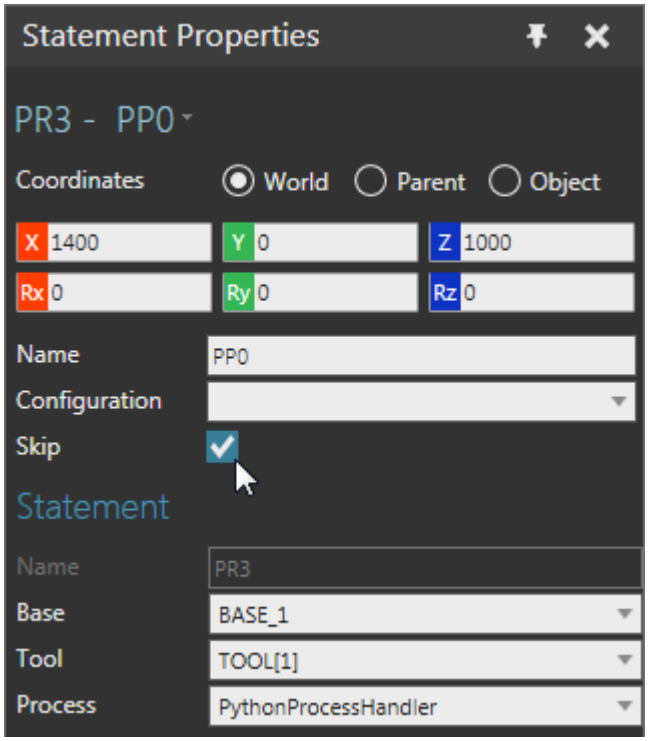
You can use the OnStatementModified event to create and update position properties. For example, you can add a property to a position to indicate whether or not the position is skipped in its process.

3. Access the PythonProcessHandler script editor, and then use the OnStatementModified event to add a Boolean type property named "Skip" to each process position, and then modify the OnStatementExecute event to only move the robot to positions that are not skipped, and then compile the code.

```
def OnStatementModified(statement):
  for pos in statement.Positions:
    skip = pos.getProperty("Skip")
    if not skip:
      pos.createProperty(VC_BOOLEAN, "Skip")


def OnStatementExecute(executor, statement):
  for pos in statement.Positions:
    if pos.Skip == False:
      executor.Controller.moveTo(pos.PositionInWorld)
```

4. In the 3D world, select **PP0**, and then in the Statement Properties panel, select the **Skip** check box.



5. Run the simulation, verify the robot skips the first position in the process, and then reset the simulation.

Review

In this tutorial you learned how to define, execute and modify a process that can be called by a robot statement. You know how to create a Python Process Handler behavior and associate it with a Robot Executor behavior. You know how to use the OnStatementAdd event to add positions and other attributes to a process statement. You know how to use the OnStatementExecute event to define the actions of a robot when executing the process statement. Finally, you know how to use the OnStatementRemove and OnStatementModifed events to clean up and modify a process statement and its positions.

For more information, see "vcPythonProcessHandler" in the Python API reference guide.