# Simio API Note: Using RunExecutable Step

April 2021

Contents

[Simio API Note: Using RunExecutable Step 1](#_Toc68549316)

[Overview 2](#_Toc68549317)

[Example 1: Python Script 2](#_Toc68549318)

[The Python Script 5](#_Toc68549319)

[Example 2: C# with Argument Logic of Delimited and None 7](#_Toc68549320)

[Under the Hood 9](#_Toc68549321)

# Overview

This API Note describes the RunExecutable Step, which is a user-defined Step that run an executable program (.exe) with optional arguments that can be specified as Simio expressions.

It has properties to perform various argument formats.

The examples all use the test folder c:\test\TestRunExecutable which is what is referenced by the example Simio project located at \Models\ModelRunExecutable.spfx

All of the example executables build to \Executables folder, so you can move the .exe files from here to the test folder. To use the Step from Simio you will also need to move the RunExecutable.DLL from the \Executables folder to your Documents\SimioUserExtensions folder.

Example 1: Python Script

The first example is running a Python script. Python is an interpreted language, so in this case the Python Interpreter is the executable, and the location of the script is the first argument.

After placing RunExecutable in the appropriate location (e.g. Documents\SimioUserExtensions), Simio desktop can be run and in the Processes tab the user-defined Step RunExecutable can be selected.

In this case we will make a process called RunPythonScript and set it as an Add-On Process Trigger for Server1 called “After Processing”:

Graphical user interface, application

Description automatically generated

The Process RunPythonScript Process is created by dragging a RunExecutable Step into the Process Edit panel. Selecting it shows its properties:

Graphical user interface, application

Description automatically generated

The Executable location is the location of the Python interpreter. In this case, it is Python version 3.8 and resides under python38 as python.exe.

Since how arguments are treated and formatted various widely, there are choices under the Property “ArgumentLogic”. Since we are going to run Python scripts we have chosen the “Python” option.

The arguments are represented as a Simio Repeating Group with two fields:

1. The name of the argument
2. The value of the argument

The value must be a Simio Expression.

Graphical user interface, application

Description automatically generated

Since this example is Python, the first argument must be a full path to the Python script.

Two other arguments Arg1 and Arg2 have been added for demonstration.

When the arguments are added to the command line, they are surrounded by double quotes so that the executable can correctly interpret any embedded spaces that may occur.

## The Python Script

When the example Python script runs, it interprets the arguments and write them to a file hardcoded to be c:\test\testRunExecutable\PythonScriptTakingArgumentsOut

Here is the Python Script:

Text

Description automatically generated

And here is the result of the Python script, which was written to c:\test\testRunExecutable\PythonScriptTakingArgumentsOutput.txt:

Text

Description automatically generated with medium confidence

# Example 2: C# with Argument Logic of Delimited and None

The second example is a C# test program that simply outputs

* The entire command line
* The arguments as C# would interpret them

We will test this program with two added Steps:

1. The exit of the Source (tests argument logic of “Delimited” with a slash)
2. The entrance of the Sink (tests argument logic of “None”)

The test executable is called TestRunArguments.exe and can be found under the GitHub Executables folder. Again, this should be moved to c:\test\testRunExecutable to harmonize with the sample Simio model ModelRunExecutable.spfx.

The output test file TestRunArguments.txt should look something like this:

A screenshot of a computer

Description automatically generated with medium confidence

# Under the Hood

All of the source code is located under the \Source folder for your convenience.

The starting of the executables employs the .NET System.Diagnostics.Process class.