Luno Tools Run Command

Description
Run Command aims to be a user-friendly way to run command line applications, command line system calls and single line shell scripts.

Setup

1. **Build a working command outside Switch**
   It is important to start with a command which achieves your task and which needs to work outside Switch.

   On Mac, this means you build and test the command in the Terminal App. On Windows, you build the command in “Command Prompt” (Cmd.exe) or “Windows Powershell”.

   By example, we can build a command which retrieves the permissions of one file and stores it in another file.
   On mac we can achieve this using the following command:
   ls -l /path/to/inputfile.txt > /path/to/outputfolder/filename.txt
   And on Windows Command Prompt using this command :
   icacls -l C:\path\to\inputfile.txt > C:\path\to\outputfolder\filename.txt

   If you have issues with your command at this stage (you don’t achieve the output you want, you have errors, …), then the issue is not related to the “Run Command” app itself. The people with the most knowledge about your issue are the people who are expert at the command line application, command line system call or shell script you are trying to use. Usually, there are help pages, support forums, support people specific for these applications who can help you. Alternatively, you can also ask help on the Enfocus forums.

2. **Handle spaces correctly**
   When working with these kinds of commands a space is used to separate different parts of the command, by example a space is placed between the command name and the path to the inputfile. This can give problems when there is a space halfway the path to the input file or a space in the filename of the input file.

   To avoid these kinds of problems you need to place the complete paths to the input and output files between quotes. This way you tell the environment that everything between quotes is one part, spaces included.

   When we apply this to our examples, then we get:
   Mac:
   ls -l “/path/to/inputfile.txt” > “/path/to/outputfolder/filename.txt”
   Windows Command prompt:
   icacls -l “C:\path\to\inputfile.txt” > “C:\path\to\outputfolder\filename.txt”
   (The commands in this stage should still work in Terminal/Command Prompt)

3. **Translate your working command to "Run Command" property settings**
   a. **Shell**
      Choose the shell you used to test the command with. If you are unsure what you need to use, use the default option. More information in the "Flow elements properties detailed info" section.
   b. **Command**
      Place the command you tested in Terminal/Command Prompt/… in this property. To make sure it works in the "Run Command" app you will have to make the following adjustments:

      Path to the input file
      The path to the input file can't be fixed, it needs to be adjusted by the app to be the path to the input file. To make this happen, replace the fixed path to the input file with %%InputFilePath%%. (Don't forget to place the quotes around it.) If your command doesn't use an input file, then it is not necessary to use %%InputPath%%.

      When we apply this to our examples, then we get:
      Mac:
      ls -l "%%InputFilePath%%" > "/path/to/outputfolder/filename.txt"
      Windows Command Prompt:
      icacls -l "%%InputFilePath%%" > "C:\path\to\outputfolder\filename.txt"

      Path to the output location
      The app will need to know where it can find the output file(s) of the command. Therefore, the "Run Command" app creates a temporary folder in which the command needs to place its output files if there are any. If you use %%OutputPath%% in the command property, then this will be replaced with the path to the folder where output files need to be placed.

      If we have a command where the output will be one file, then we can build the output path like this:
      "%%OutputPath%%\filename.extension"

      If you wish to keep the same filename proper as the input file, then you can use Switch variables like this:
      "%%OutputPath%%\[Job.NameProper].extension"

      If the output file will have the same name and extension of the input file, then you can use this:
      "%%OutputPath%%\[Job.Name]"

      The current directory in which the command is executed is also the folder where the app expects the output files. This means you can use a relative path instead of an absolute path to shorten the path to the output location.

"%%OutputPath%%\[Job.NameProper].extension"
can be shortened to:
".\[Job.NameProper].extension"
or even:
"[Job.NameProper].extension"

"%%OutputPath%%\[Job.Name]"
can be shortened to:
".\[Job.Name]"
or even:
"[Job.Name]"

When we apply this to our examples, then we get:
Mac:
ls -l "%%InputFilePath%%" > "[Job.NameProper].txt"
Windows Command Prompt:
icacls -l "%%InputFilePath%%" > "[Job.NameProper].txt"

This documentation is created using Microsoft Word which does some automatic changes to quotes to make them look nice. These automatic changes can cause issues when these double quotes are used in a command. Therefore, please make sure to remove and re-add the double quotes from the above command when you test it in Switch.

c. **Private data tags for standard output, error output and exit code.**
These settings will usually keep their default values.

Depending on what your command does it may show some text in the Terminal/Command Prompt/… window.

The text you see is actually a combination of two kinds of output of the command: "Standard output" and "Error output". "Standard output" (sometimes abbreviated to stdout) is the usual informative output. If something in the command went wrong, then the application will give an indication of what went wrong in "Error output" (sometimes abbreviated to stderr).

The information provided by the started application via standard output and error output is available as private data in the output job. If the command generates information that you would like to use in your flow, then you can use the standard output information which is stored in the private data tag.

Standard output and error output are good indicators if a command was successful or not. However, they are not always available. Next to these two kinds of output a command will always give another piece of output: the exit code. If the exit code is 0, then this means that the command was successful. If the exit code is something different, then something went wrong. You can usually find which exit code corresponds to which problem in the documentation of the application.

If the command is incorrect in a way which causes that the application can't be started, then you will receive an exit code and error output from the shell you are using instead of the command itself. If you see exit codes which are not in the documentation of the application, then you are most likely in this situation.

(The above explanation is assuming that the application/script you are starting utilizes standard output, error output and exit code the way they are intended. Some scripts you find on the internet may send errors to standard output instead of error output or divert from the above explanation in another way.)

4. **Test the settings and check your output**
   Test the configuration of the "Run Command" app using some test files to make sure it runs correctly. If it doesn't, make sure to check the Switch messages.

   If the command provided an exit code which is not 0 or the command provided some information on error output, then the output will be sent over the error out connection. In this case the actual executed command, the exit code and error output will be logged in the Switch messages.

   If the command gives an exit code which is 0 and no information on error output, then the output will be sent over to the success out connection. The messages will contain a truncated version of standard output. If you wish to see the full standard output, please enable debug logging and check the debug messages.

   If there is no outgoing error connection and jobs are sent to such a connection, then these jobs will be removed from the flow. This is a limitation in the current scripting API, so I can't avoid it as an app developer. **Therefore, please make sure there always is an error out connection to avoid that jobs get lost.**

   If there is output in the folder intended for output ("%%OutputPath%%), then these file(s) will be sent to the outgoing connection. If there are multiple files/folders, then they will be combined in one folder when they are sent to the outgoing connection.

   If there is no output in this folder, then the input job will be sent to the outgoing connection.

Changing file permissions of jobs
Due to limitations in the current Node.js scripting API in Enfocus Switch it is not possible to use "Run Command" directly on job files when running a command to change file permissions. Instead, it is necessary to copy the file to a temporary location (like C:\Temp), change the permissions on the file in that location and then inject that file into the flow. There is a sample flow to illustrate how you can build a flow like this.

Compatibility
Switch 2021 Spring and higher.

Flow elements properties detailed info

- **Shell**
  Here you can choose in which shell the command needs to be executed. If you are unsure what to choose, pick the option listed as default.

  Mac: Bash or Zsh
  On Mac you are usually using Apple Terminal to test your commands. In older versions of OSX bash was the default shell used by Terminal, starting from OSX Catalina Zsh is the default shell used in Apple Terminal. The difference between both in situations relevant for the “Run Command” App should be very limited.

  Windows: Command Prompt or Powershell
  If you are using the black window with white letters to type your commands, then you are using Command Prompt. If you are using the blue window, then you are using Powershell.

  Powershell Core
  Powershell core is only available when it is installed on the system running Switch. Mac and Windows installers can be downloaded on the Powershell github page. It is possible that the system needs to be rebooted before Powershell Core is available in the dropdown list.

  When you are using Powershell or Powershell Core to start a command line application (instead of a native Powershell command), please don’t forget to start the command with “& “.

- **Command**
  This is the command which will need to be executed.
  %%InputFilePath%% will be replaced with the path to the input file.
  %%OutputPath%% will be replaced with the path to the folder in which output files need to be placed.
  For more information, please check the setup section of the documentation.

- **Standard output private data tag**
  The standard output (stdout) provided by the started application is stored in private data. This property sets the tag of the private data.

- **Error output private data tag**
  The error output (stderr) provided by the started application is stored in private data. This property sets the tag of the private data.

- **Exit code private data tag**
  The exit code provided by the started application is stored in private data. This property sets the tag of the private data.

Differences compared to the Execute command tool

- **Less properties to simplify setup**

- **Shell**
  In execute command you are executing direct operating system commands. When you are running commands in Terminal/Command Prompt, then you are executing commands in a shell. This causes a difference in behavior which can be confusing. Therefore, commands are executed in a shell in the “Run Command” app.
  As a result, you can use things like redirecting and piping without having to rely on intermediate shell scripts.

- **Concurrent vs serialized**
  The execute command tool has a property to easily set if the task can run concurrently (multiple times at the same time) or serialized (only one task at the same time). The “Run Command” app runs concurrently by default. If you wish to run a command serialized, then please enable Advanced Performance Tuning in the properties of the flow and then simulate serialized processing by setting “Number of slots” to 1.

- **Standard output and Error output in Private Data instead of datasets**
  Storing the textual output in private data makes them very easily accessible. However, if you are using the “Run Command” app and you need to have the textual output in a file to store it as a dataset, then it will be necessary to redirect the output to a file in your command. This way you can use one of the metadata pickup tools in Switch to pickup the file to make it a Switch dataset.

- **Workaround necessary to change permissions of job files.**
  More info in section “Changing file permissions of jobs”