New issue   **Jump to bottom**

[Core] Add `az.ps1` entry script for PowerShell #23514

⑄ **Merged**   *jiasli* merged 2 commits into **Azure:dev** from **jiasli:ps1** ⧉ on Aug 19

Conversation 9   Commits 2   Checks 48   Files changed 4

👤 **jiasli** commented on Aug 11 • edited ▾   Member

Context

Fix #20972

If an argument contains special characters like `|` and `&`, even though the character is double quoted, it still gets executed. This is because PowerShell launches `cmd.exe` to execute `az.cmd` script. Double quotes are parsed by PowerShell so not passed to `cmd.exe` during this process. `cmd.exe` sees the raw `|` and `&` and executes them.

However, when `az.cmd` is invoked from `cmd.exe` (Command Prompt), it is launched in-process, and double quotes are preserved, leading to inconsistent behavior between `cmd.exe` and PowerShell.

Change

This PR adds `az.ps1` entry script for PowerShell, so that PowerShell can natively execute `az.ps1` and no longer launches `cmd.exe` to execute `az.cmd`.

Since PowerShell 7.3, double quotes are now preserved correctly when calling a native executable (PowerShell/PowerShell#1995 (comment)), thus fixing #15529.

Testing Guide

```
az --debug '"ab"'
cli.knack.cli: Command arguments: ['--debug', '"ab"']

az --debug "a|b"
cli.knack.cli: Command arguments: ['--debug', 'a|b']
```

```
az --debug "a&b"
cli.knack.cli: Command arguments: ['--debug', 'a&b']
```

👁 **SteveL-MSFT** reviewed on Aug 11

View changes

```
build_scripts/windows/scripts/az.ps1
```

...	...	@@ -0,0 +1,2 @@
	1	+ $env:AZ_INSTALLER="MSI"
	2	+ & "$PSScriptRoot\..\python.exe" -IBm azure.cli $args

SteveL-MSFT on Aug 11 • edited ▾

This doesn't work correctly with WinPS 5.1, right? Unfortunately the case of `az '{"test":1}'` by the time the `$args` is processed, you would get `{test:1}` on any PS older than 7.3 (which makes it no longer valid JSON), and there's no way to know what the original intent. So perhaps the best thing is to iterate through `$args` and if `|` or `&` is found, throw an error that it would only work with 7.3+?

jiasli on Aug 11   Member   Author

Tested in Windows PowerShell 5.1, and indeed double quotes are lost, but luckily `|` or `&` isn't causing any problems to us. :)

```
> az2 --debug "a|b" "a&b" '"ab"'
cli.knack.cli: Command arguments: ['--debug', 'a|b', 'a&b', 'ab']
```

👁 **yonzhan** requested a review from **evelyn-ys** 4 months ago

👤 **yonzhan** assigned **jiasli** on Aug 11

yonzhan commented on Aug 11   Collaborator

add a ps1 file as the entry script to bypass 'PowerShell -> cmd' issue.

🏳 **yonzhan** added this to the **Aug 2022 (2022-09-06)** milestone on Aug 11

jiasli commented on Aug 12   `Member` `Author`

By the way, I am playing around by calling `cmd.exe` (https://ss64.com/nt/cmd.html) myself from Command Prompt and it looks like it has some weird behavior that I need to add extra double quotes (denoted by `^`):

According to https://ss64.com/nt/syntax-esc.html

> To launch a batch script with spaces in the script Path and other parameters, all requiring quotes:
>
> ```
> CMD /k ""c:\batch files\test.cmd" "Parameter 1 with space" "Parameter2 with space""
>        ^                                                                           ^
> ```

My testing:

`D:\test.cmd` contains

```
@echo %*
```

```
D:\>cmd /c "D:\test.cmd" "parameter1 parameter2"
'D:\test.cmd" "parameter1' is not recognized as an internal or external command,
operable program or batch file.

D:\>cmd /c ""D:\test.cmd" "parameter1 parameter2""
"parameter1 parameter2"
```

While in Bash

`/home/user2/test.sh` contains

```
echo "$@"
```

```
user2@DESKTOP-A79F1:~$ bash "/home/user2/test.sh" "parameter1 parameter2"
parameter1 parameter2
user2@DESKTOP-A79F1:~$ bash ""/home/user2/test.sh" "parameter1 parameter2""
bash: /home/user2/test.sh parameter1: No such file or directory
```

No weird outer double quotes are needed.

↰ 🧑 **jiasli** force-pushed the `ps1` branch from **e0d4146** to **d488563** 3 months ago   `Compare`

-○- 🧑 **ps1**   ✓ 61a4b0d

jiasli force-pushed the `ps1` branch from **1b50679** to **61a4b0d** 3 months ago

Compare

jiasli marked this pull request as ready for review 3 months ago

`pip`

✓ 110fd2c

evelyn-ys approved these changes on Aug 17

View changes

bebound approved these changes on Aug 19

View changes

jiasli changed the title ~~[Misc.] Add `az.ps1` entry script for PowerShell~~ [Core] Add `az.ps1` entry script for PowerShell on Aug 19

jiasli merged commit **a55f924** into `Azure:dev` on Aug 19
50 checks passed

View details

jiasli deleted the `ps1` branch 3 months ago

---

jiasli commented on Aug 19   Member   Author

I also found another weird PowerShell behavior while writing this PR - PowerShell sometimes even depends on the space location to determine the behavior:

```
> python -c "import sys; print(sys.argv)" '\"a b\"c'
['-c', '"a', 'b"c']
> python -c "import sys; print(sys.argv)" '\"ab\" c'
['-c', '"ab" c']
```

Also consider these commands:

```
> python -c "import sys; print(sys.argv)" '""a b""'
['-c', '"a b"']
> python -c "import sys; print(sys.argv)" '\"a b\"'
['-c', '"a', 'b"']
```

They should be grammatically equivalent, as in Win32 API within double quotes, `""` and `\"` both mean literal double quote sign. However, they are not, because in the second example, PowerShell somehow recognizes double quotes and thinks space is not "on the root level of the string", so doesn't surround the argument with double quotes when passing it to `python.exe`, making the argument split.

```
> python -c "import sys; print(sys.argv)" '\"a b\" '
['-c', '"a b" ']
```

In the above example, I put an extra space "on the root level" and PowerShell passed the arguments double-quoted-ly to `python.exe`.

👁 **jiasli** commented on Aug 20

View changes

src/azure-cli/az.ps1

```
 1   + $env:AZ_INSTALLER="PIP"
 2   +
 3   + if (Test-Path "$PSScriptRoot\python.exe") {
 4   +     # Perfer python.exe in venv
 5   +     & "$PSScriptRoot\python.exe" -m azure.cli $args
 6   + }
 7   + else {
 8   +     # Run system python.exe
 9   +     python.exe -m azure.cli $args
10   + }
11   +
12   + # SIG # Begin signature block
13   + # MIInqgYJKoZIhvcNAQcCoIInmzCCJ5cCAQExDzANBglghkgBZQMEAgEFADB5Bgor
14   + # BgEEAYI3AgEEoGswaTA0BgorBgEEAYI3AgEeMCYCAwEAAAQQH8w7YFlLCE63JNLG
```
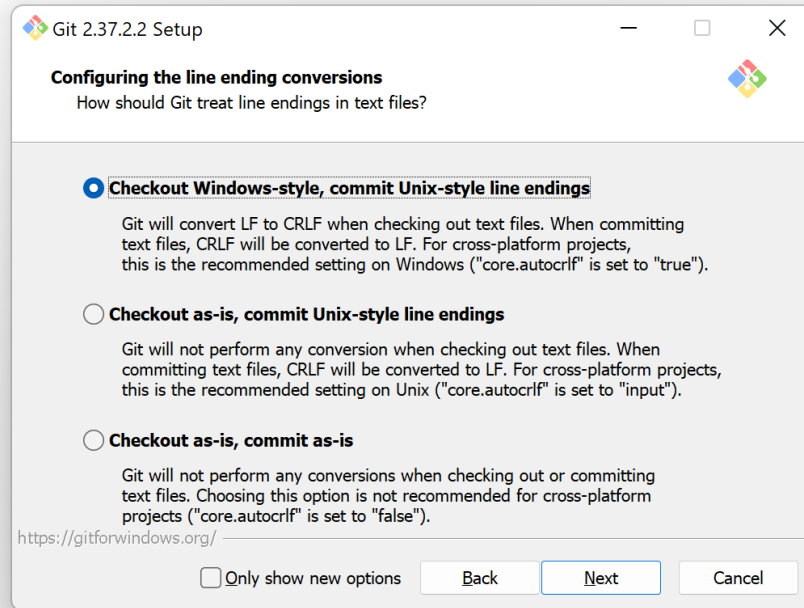
🧑 **jiasli** on Aug 20 • edited ▾   Member   Author

There is one problem with `az.ps1` for `pip`.

- ESRP code-sign was executed on Azure DevOps's Windows agent. When `git` checks out the source code on Windows, the end of line sequence is `CRLF` (`\r\n`). This is the default of Windows `git`:

so the signature was generated for the `CRLF` version of `az.ps1` .

- However, wheels are built on Linux agent, converting `az.ps1` 's end of line sequence to `LF` (`\n`).
- Even when installing with `pip -e` , `pip` converts `CRLF` to `LF` (for simplicity I guess).

That is, on Windows,

- `az.ps1` in source code contains `CRLF`
- `az.ps1` installed in `Scripts` folder contains `LF`

Since the signature was generated for the `CRLF` version of `az.ps1` , but the installed `az.ps1` is `LF` version, the signature doesn't match the file content.

Even if we ESRP code sign the `LF` version, `az.ps1` will be checked out as `CRLF` in the source code, still making the signature mismatch. That is, the signature will never match the file content of both `az.ps1` s - one `az.ps1` 's signature must be out of place.

↗ This was referenced on Sep 7

Cannot pass PowerShell array as `$mylist` to Azure CLI #23797

⊘ Closed

{Core} Entry script: Use `@args` to tolerate incorrectly passing list with `$` #23802

⥯ Closed

↗ 🌐 **AzureMarker** mentioned this pull request on Sep 14

Azure CLI 2.40.0 breaks stdin passing on PowerShell #23883

⊙ Open

krokofant mentioned this pull request on Sep 13

Getting "Unable to parse parameter" with "az deployment group create" and parameter files since 2.40.0 #23854

⊘ Closed

mjhilton mentioned this pull request on Sep 18

Azure CLI - Run a Script - Deployment Stuck at Login with Service Principal
OctopusDeploy/Issues#7782

⊘ Closed

⊟ 1 task

FISHMANPET mentioned this pull request on Sep 20

PSAvoidUsingPositionalParameters reports issue with az command when version 2.40.0 of az cli is installed PowerShell/PSScriptAnalyzer#1845

⊘ Closed

☐ This was referenced on Sep 26

`az` can't be invoked by `Start-Process` since 2.40.0 #24005

⊙ Open

[Core] Revert #23514: Rename entry script `az.ps1` to `azps.ps1` #24015

⌥ Merged

az command returns 0 when failing #23880

⊘ Closed

☐ jiasli added a commit that referenced this pull request on Sep 30

    [Core] Revert #23514: Rename entry script az.ps1 to azps.ps1 (#24015 ⋯   ✓ 2f309cf

a3code commented on Oct 4

Hello everyone, could please explain how I should run scripts right now

I am a little bit confused, on azure DevOps I have a script that had run az in the next way:

```
Start-Process "az" -ArgumentList $azArgs -NoNewWindow -Wait
```

azArgs is a commend to set some settings that is.

and currently I see an error :

Start-Process : This command cannot be run due to the error: %1 is not a valid Win32 application

how do I need to change it to make it work after this fix?

Should I just remove Start-Process ?

---

yonzhan commented on Oct 4   Collaborator

We will revert this script and roll out on 10.12. Everything will work normally by then.

☑ 🧑 **jiasli** mentioned this pull request on Oct 6

PowerShell stop-parsing token `--%` stops working since Azure CLI 2.40.0 #24114

⊘ Closed

☑ 🧑 **jiasli** mentioned this pull request on Oct 16

az ad group: Group names with '&' in them get split at the point where the '&' appears
#24209

⊘ Closed

---

ishepherd commented on Oct 18

@jiasli @yonzhan Will you be adding test coverage for all the many issues people are reporting? Looks like a nice opportunity to fill coverage gaps.

Reviewers

🧑 SteveL-MSFT   💬

🧑 bebound   ✔

🧑 evelyn-ys   ✔

Assignees

🧑 jiasli

Labels

None yet

Projects

None yet

Milestone

Aug 2022 (2022-09-06)

Development

Successfully merging this pull request may close these issues.

⊘ **Issue to set password for secret at KeyVauklt by azure CLI**

---

7 participants