Referencing Launch Arguments in Scripts

Description

Scripts ran through FileWave have the option to supply 'Launch Arguments'. These are referenced from the script but are not included in the body of the script.

They may be supplied to any of the following:

- Fileset Scripts
- Custom Fields
- Policy Blocker Scripts

Often Admins feel that there is a limit of 9 'Launch Arguments' through FileWave, but the below will demonstrate this is not the case.

Information

The Launch Arguments, known as Positional Parameters, are referenced as follows:

	macOS/Linux	Windows Powershell	Windows Bat
First Argument	$1	`$args[0]`	%1
Second Argument	$2	`$args[1]`	%2
Third Argument	$3	`$args[2]`	%3

More may be added and each is referenced in turn by its positional place as in the above table.

Considerations

Certain shell types behave differently. This may particularly show when referencing the 10th or higher supplied argument.

> Although two solutions have been supplied, zsh is recommended to stay in line with Apple's policy: https://support.apple.com/en-gb/HT208050

bash and sh

The following example was hoped to print the first 3 positional parameters, followed by the 10th and 11th.

bash_test.sh

```
#!/bin/bash

echo $1
echo $2
echo $3
echo $10
echo $11

exit 0
```

However, if the character '1' was supplied as single argument the following would be observed when ran:

bash_test.sh

```
./bash_test.sh 1
1


10
11
```

The bash and sh shells are examples which treat $10 as ${1}0, $11 as ${1}1, etc.; returning the value of $1 and then appending the

additional character (0 or 1 in this example).

Additional reference:

- https://www.oreilly.com/library/view/bash-cookbook/0596526784/ch05s07.html
- https://www.oreilly.com/library/view/bash-cookbook/0596526784/ch05s04.html

As such the above output is equivalent to:

	Description	Output
$1	Returns first argument	1
$2	Returns second argument	2
$3	Returns nothing, no third argument	
$10	Returns first argument, followed by the '0' character	10
$11	Returns first argument, followed by the '1' character	11

To ensure the correct positional parameters are referenced, the variables must be explicitly set to achieve the correct output.

The following shows the corrected script.

bash_test.sh

```
#!/bin/bash

echo $1
echo $2
echo $3
echo ${10}
echo ${11}

exit 0
```

Now when executed with 11 positional parameters, the expected output is displayed:

bash_test.sh

```
./bash_test.sh var1a var2b var3c var4d var5e var6f var7g var8h var9i var10j var11k
var1a
var2b
var3c
var10j
var11k
```

zsh

Not all shells act the same. An alternate to the above would be zsh. Taking the above example as a zsh script:

zsh_test.sh

```
#!/bin/zsh

echo $1
echo $2
echo $3
echo $10
echo ${11}

exit 0
```

Given the same 11 arguments, this would output:

zsh_test.sh

```
./zsh_test.sh var1a var2b var3c var4d var5e var6f var7g var8h var9i var10j var11k
var1a
var2b
```

```
var3c
var10j
var11k
```

Unlike bash and sh, zsh considers $10 to be the 10th argument, $11 the 11th argument and so on and so forth. Notice zsh may use either format.

Conclusion

Often admins are confused with output results when using 10 or more arguments. Armed with the above knowledge, this should assist structuring scripts.

Related Content

- Custom Fields
- Filesets / Payloads

Revision #1
★Created 2 July 2023 14:58:05 by Josh Levitsky
✎Updated 2 July 2023 15:03:31 by Josh Levitsky