

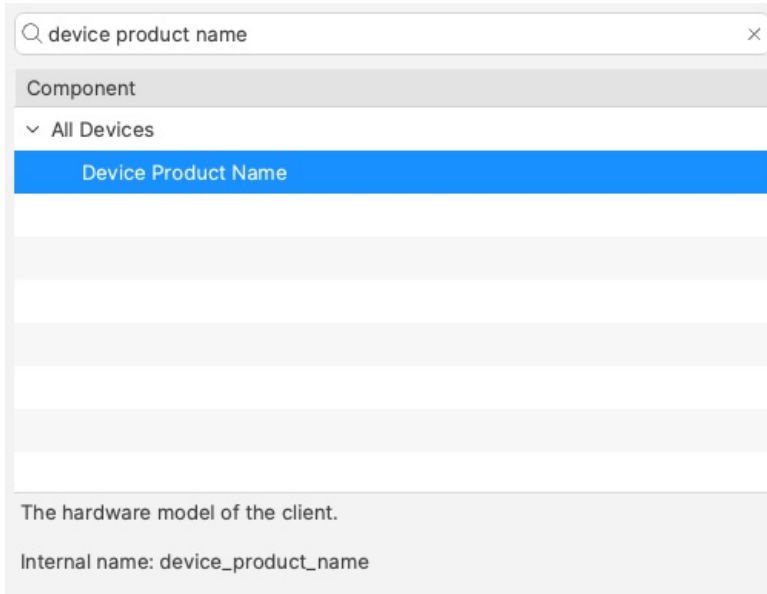
# Inventory Items in Scripts

## What

- Each Inventory Items has an Internal Name, including Custom Fields which provide extended inventory
- The Internal Name can be used to reference an Inventory Item in Scripts
- These Internal Names should be added to either the Launch Arguments or Environment Variables of the Script
- This applies to all script types, be that other Custom Fields, Policy Blocker Scripts or Fileset Scripts

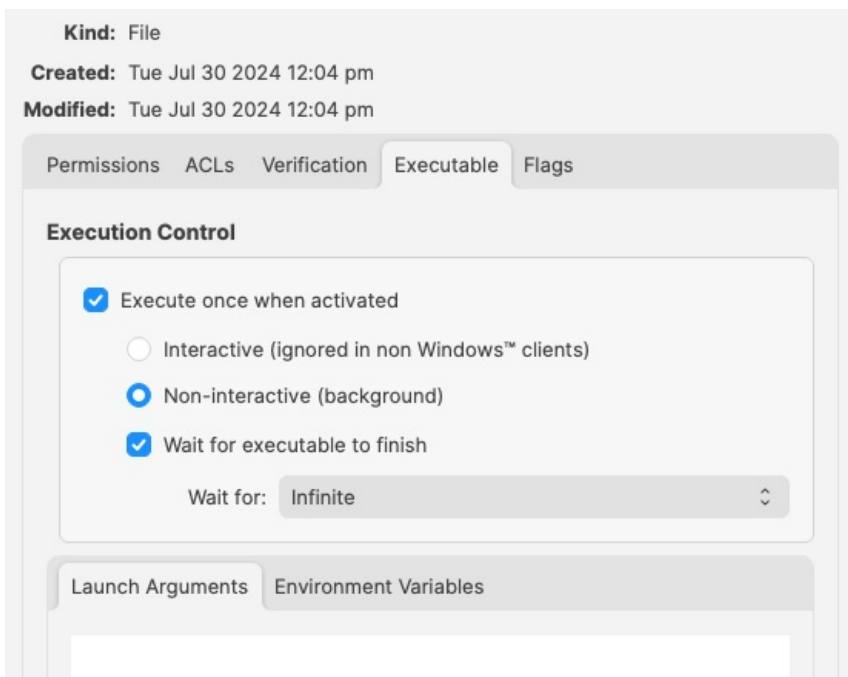
## When

Internal Name of an Inventory Item may be located from the Inventory Query Editor. Example shows the Internal Name: 'device\_product\_name'



The screenshot shows a search bar at the top with the text 'device product name'. Below it is a table with a header 'Component' and a row 'All Devices'. Under 'All Devices', there is a row 'Device Product Name' which is highlighted in blue. Below the table, there is a text box containing the description 'The hardware model of the client.' and the internal name 'Internal name: device\_product\_name'.

This may then be added into a Script, by way of either a Launch Argument or Environment Variable



The screenshot shows a window for configuring a script. At the top, it says 'Kind: File'. Below that, it shows 'Created: Tue Jul 30 2024 12:04 pm' and 'Modified: Tue Jul 30 2024 12:04 pm'. There are four tabs: 'Permissions', 'ACLs', 'Verification', 'Executable', and 'Flags'. The 'Executable' tab is selected. Under the 'Executable' tab, there is a section 'Execution Control' with the following options: 'Execute once when activated' (checked), 'Interactive (ignored in non Windows™ clients)' (unchecked), 'Non-interactive (background)' (checked), and 'Wait for executable to finish' (checked). Below these options, there is a 'Wait for:' field with the value 'Infinite'. At the bottom, there are two tabs: 'Launch Arguments' and 'Environment Variables'.

But, which should be used?

## How

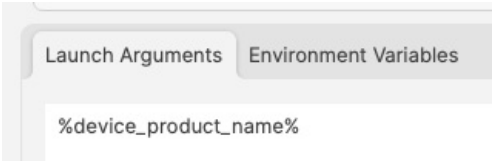
In some respects it does not matter which is used, however, for easy reference consider the following:

- Launch Arguments are referenced by their numerical position
- Environment Variables are referenced by a chosen name
- Custom Fields have an abbreviated name and a full name. Custom Field names could overlap with a built-in Inventory Item.

## Built-In Inventory

In general, recommendation here is that of Environment Variables. This makes reading the script easier without having to redefine new names within the script for Launch Argument positions.

For example:



Could be referenced in a script as:

macOS shell	echo \$1
Windows Powershell	echo \$args[0]

But to make the parameters more easily recognisable for anyone reading the script, it could be desirable to name them:

macOS shell	<div>product_name="\$1" echo \$product_name</div>
Windows Powershell	<div>\$product_name="\$args[0]" echo \$product_name</div>

References to the provided inventory parameters in the script now makes more sense, but as mentioned, Environment Variables take this a step further:



A variable name is already defined and this can be referenced in the script directly

```
echo $product_name
```

## Improvements

### First Improvement

To improve the readability of the script further, consider setting the variable name to match the value, e.g:



```
echo $device_product_name
```

### Second Improvement

When referencing a Custom Field in a script, it could be referenced in one of two ways.

### Field Details

**Name**

State

**Internal Name**

Using internal name the field can be referenced in other parts of FileWave

state

**Description**

Custom Field

✔ Note, the description has been used to indicate this is a Custom Field. Inventory Query editor shows Description.

This could be referenced with:

Launch Arguments		Environment Variables	
Variable		Value	
state		%state%	

and

```
echo $state
```

However, there is a built-in Inventory Item called State. So there are now two Internal Names of 'state'

**Custom Field**

Internal name: state

Device state - Tracked, Archived, Missing, Untracked, Disabled.

Internal name: state

The above scripted example for 'state' would actually report the built-in value, not the Custom Field. There is, though, a hidden prefix that can be used.

This Custom Field could be referenced as either:

- %state%
- %CustomFields.state%

The latter prevents unexpected collusion with the matching Internal Name. Hence, to make the parameters more obvious when reading...

Launch Arguments		Environment Variables	
Variable		Value	
custom_fields_state		%CustomFields.state%	
internal_device_product_name		%device_product_name%	

```
echo $custom_fields_state
echo $internal_device_product_name
```

📘 Notice, despite no prefix existing for built-in Inventory Items, by including a prefix for both variables in the Environment Variables definitions, reading the script will be much clearer.



Anyone reading the script is now aware that state is a Custom Field, without having to cross reference anything. Likewise, the reader also is aware that the device\_product\_name also comes from Inventory, again, without any cross reference necessary.

## Unknown Inventory

Not all Inventory Items are available as parameters.



The FileWave Client builds out the report of items to inventory and return to server. Additionally, all Custom Fields, including those server-side (Administrator Custom Fields), are available to the client. However, inventory returned by MDM is not available, since the client is unaware of these values, they are pure server-side.

%CustomFields.location%

---

🔄Revision #1

★Created 30 July 2024 22:49:25 by Sean Holden

✎Updated 30 July 2024 23:04:52 by Sean Holden