Custom Fields

Custom Fields

Added in FileWave 12.7.0, Custom Fields will allow you to create custom inventory values and assign them to your devices in a 5 different ways. You will find this option under the Assistants menu called Custom Fields, from there you can either Import CSV or Edit Custom Fields. The Edit Custom Fields section has four different options you can chose from that will allow you to create custom fields and in turn assign those values to devices.

Important Note: You cannot use special characters in the creation of Custom Fields!

See Importing and Exporting Custom Field Files

Display Name

A user friendly name for your reference later

Internal Name

A system-wide variable that can be used in to reference these field values

FileWave will let you create an internal name which is the same as a FileWave inventory item internal name, for example %location% is already a built-in variable. When you are referencing custom values prefix them as below to avoid conflicts.

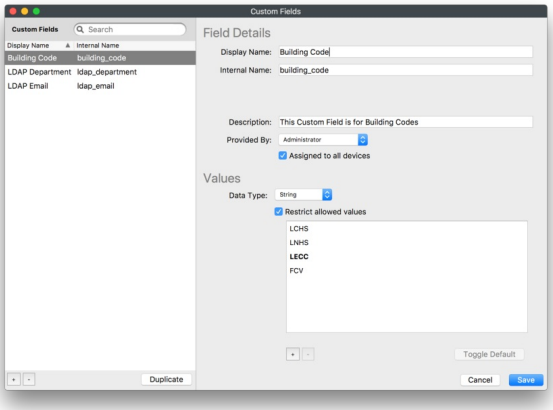
Method	Prefix	Example
MDM Profile	custom_field	%custom_field.location%
Scripted (Launch Argument or Environment Variable)	CustomFields	%CustomFields.location%

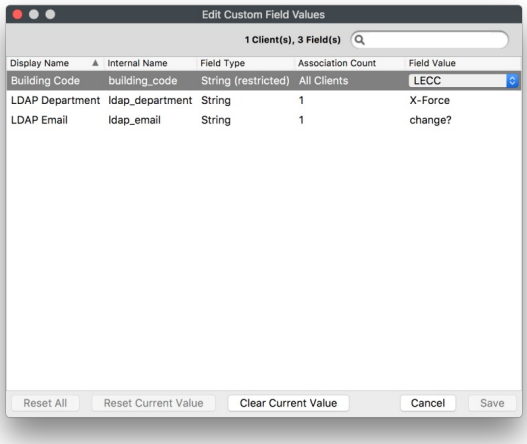
Provided by

This defines where the field comes from and also how it is updated.

Administrator

This option will allow you to set a custom field to your selected device(s) with a single or multiple preset values. This will be admin data so nothing has to be sent out to the device themselves. The setting "Assigned to all devices " will auto-associate this custom field to all devices in FileWave and any new devices that become enrolled. After picking the Data Type, you can select either a Default Value or if you would like to Restrict Allowed Values. If "Restrict allowed values" is selected then you can enter as many values as you like and then select one as default. If no value is selected as default then the value will simply be empty for the device. To modify or add these values simply right click on your selected device, select Edit Custom Field(s) Value, in the Field Value column double click and select the value from your list.





Client Script

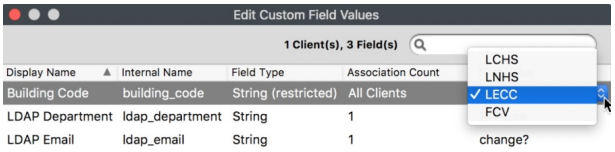
Client Script will allow you to create and send out a script to associated devices. The output of this script will be what's written as the value for your custom field. You also have the option to set the output of the script as the custom field value only if the script has the exit code of 0. This option is for both macOS and Windows using the following script types:

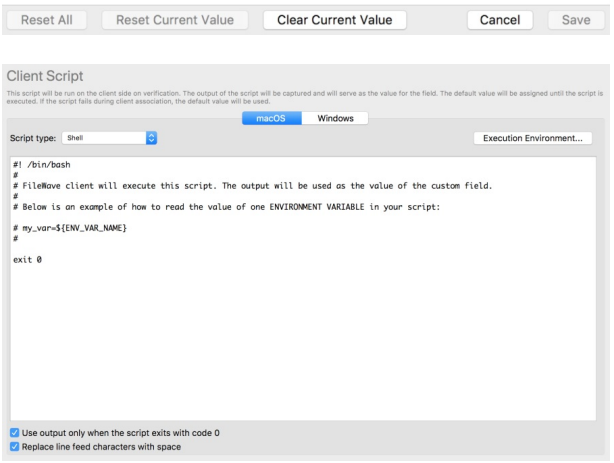
- Shell
- Python
- Perl
- Bat
- PowerShell

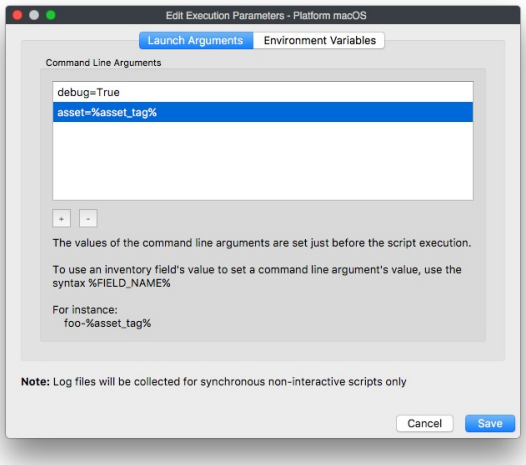
> ⚠ Important: Python and Perl will need to be installed on your Windows clients. When installing Python and Perl make sure a system path is added to environment variables during install.

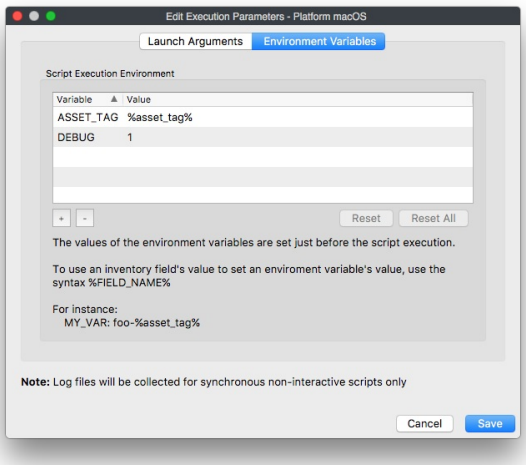
The Script will be executed after each verify, before sending inventory data.

Another option you have for script, is setting launch arguments and Environment Variables. (This option is also for any scripts you have in Desktop Filesets) So now any inventory field value can be plugged into a script.









> Dates must be ISO-8601 format (for instance: 2011-08-27T23:22:37Z).

The last line feed will be ignored to ease conversion, as commands like "echo" (bash) or "print" (python) add a line feed at the end of standard output. Scripts are stored encrypted on the disk and the FileWave client will automatically decrypt them when it needs to run. The encryption used is RSA 2048 bits, with no symmetric key exchange (only RSA).

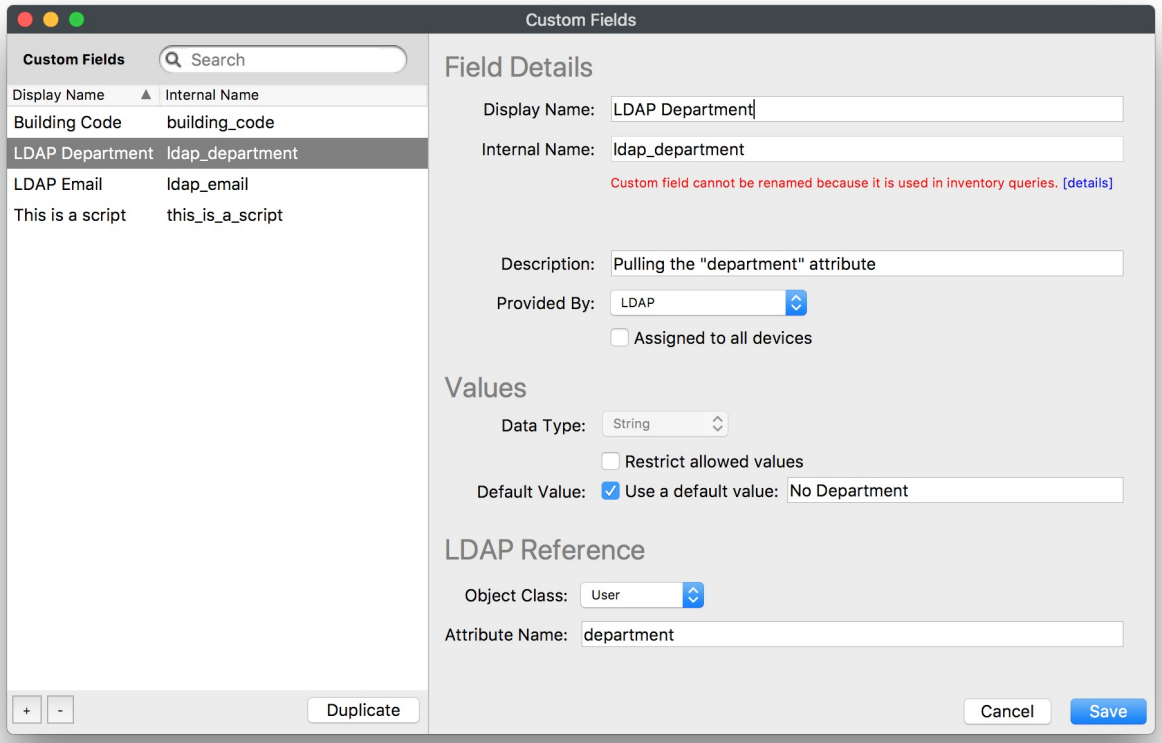
Client Command Line

This is used for existing Custom Fields that were made prior to FileWave 12.7.0 using a script to write to the custom.ini file on the client, this generally won't be used going forward for new custom fields. But if you have used custom queries with FileWave before, you will notice all of those are still present in the Custom Fields window. This will also allow you to now change the names from the deafult "custom_string_01" name for instance, to whatever you like.

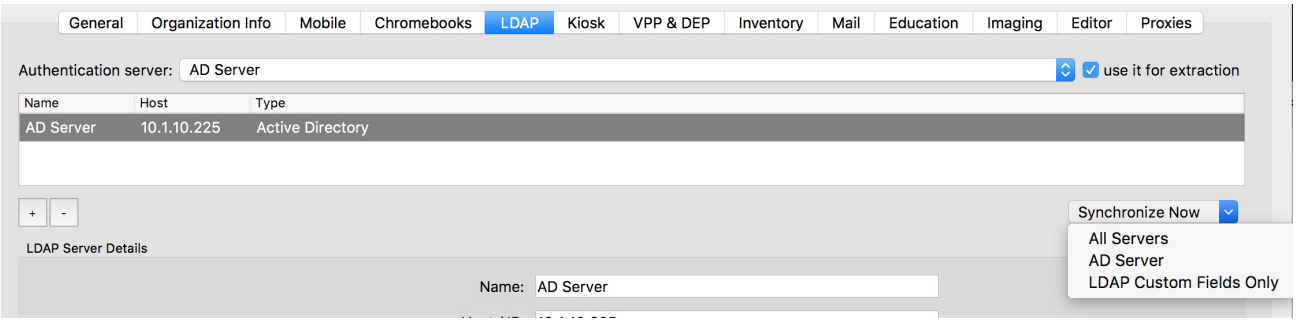
LDAP

You will be able to pull attribute values from whichever directory service is being extracted in the LDAP tab in FileWave Preferences. These values are then assigned to your devices in FileWave so that you can query them at anytime. You will simply find the attribute you would like to query such as "department", which in this case is a String type. Your chooses are String, Integer, Boolean, Date/Time. If the value does not match the data type you will get a type conversion error flag when the value is pulled. Then the object class which is either user or computer:

- User: LDAP entry is matched using either Authusername for iPads or the last LDAP user to log into a macOS and Windows device.
- Computer: LDAP entry is found using the device name in inventory against the computer name in the LDAP directory.



_How often does LDAP get scanned for updated values?
_Anytime a custom field is assigned to a device in FileWave or when the LDAP server is synced in the FileWave Preferences (this is either at the Refresh interval you can set or manually). However if a LDAP Custom Field is modified, your directory service will not be scanned right away, instead it will be scheduled to scan in 120 seconds; which is the minimum. (to change this time please contact FileWave support).



Other LDAP Considerations:
If the value of the attribute you specified is empty or the attribute is not found in LDAP, then the value of the Custom Field will simply be empty.

In the case of an attribute that has multiple entries, all entries will be returned as encoded JSON array for string custom fields. For other types of custom fields the value would contain the type conversion error flag instead. The order of entries in the JSON array is not specified.

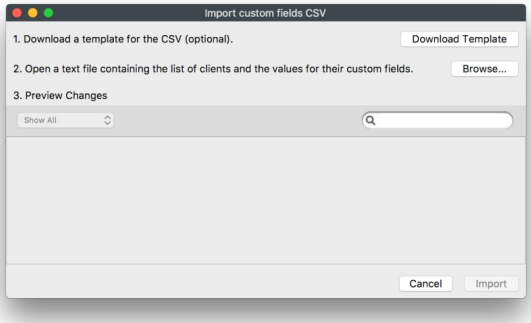
It may happen where no matching LDAP user or computer is found for a given client associated with LDAP custom field(s). In this case, the appropriate status for custom fields values will be set upon extraction ("Matching LDAP User/Computer Not Found"). Administrator has a global option to clear current custom field(s) value when such situation occurs. The option can be found in Preferences, Inventory tab, "Clear value if there is no match between client and LDAP user or computer" checkbox.

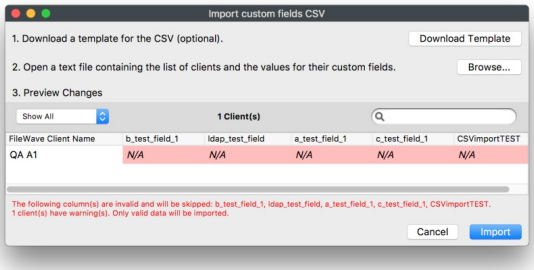
Importing from a CSV

This will allow you to change values of Custom Fields that have already been associated to devices in bulk. In the Import Custom Fields CSV window we provide a template that will let you choose not only which existing custom fields you would like in the template, but also how you would like to identify devices. Identifying devices in this section can be done with FileWave Client Name, Serial Number, Device ID, or FileWave ID.

It is important to remember that the only custom field values that will be changed are ones that have already been associated in FileWave to your devices. If you upload a CSV that specifies a value for a device that does not have the corresponding

custom fields associate to it prior to upload, then you will see an error telling you those values were skipped.





Related Content

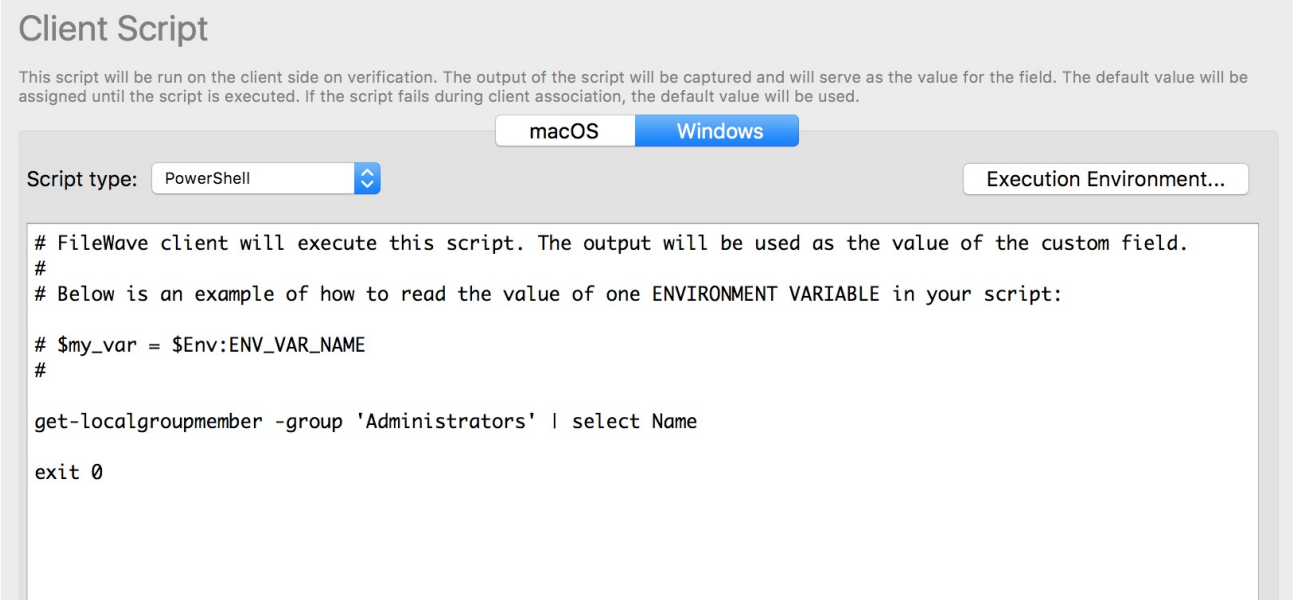
- Importing and Exporting Custom Field Files

Running Built-in PowerShell Commands with Custom Fields

In Filewave when using custom fields for PowerShell scripts the commands are run in a 32-bit environment. Some built-in PowerShell commands require 64-bit to work. In order to run the commands you need to modify the command to launch the correct PowerShell environment.

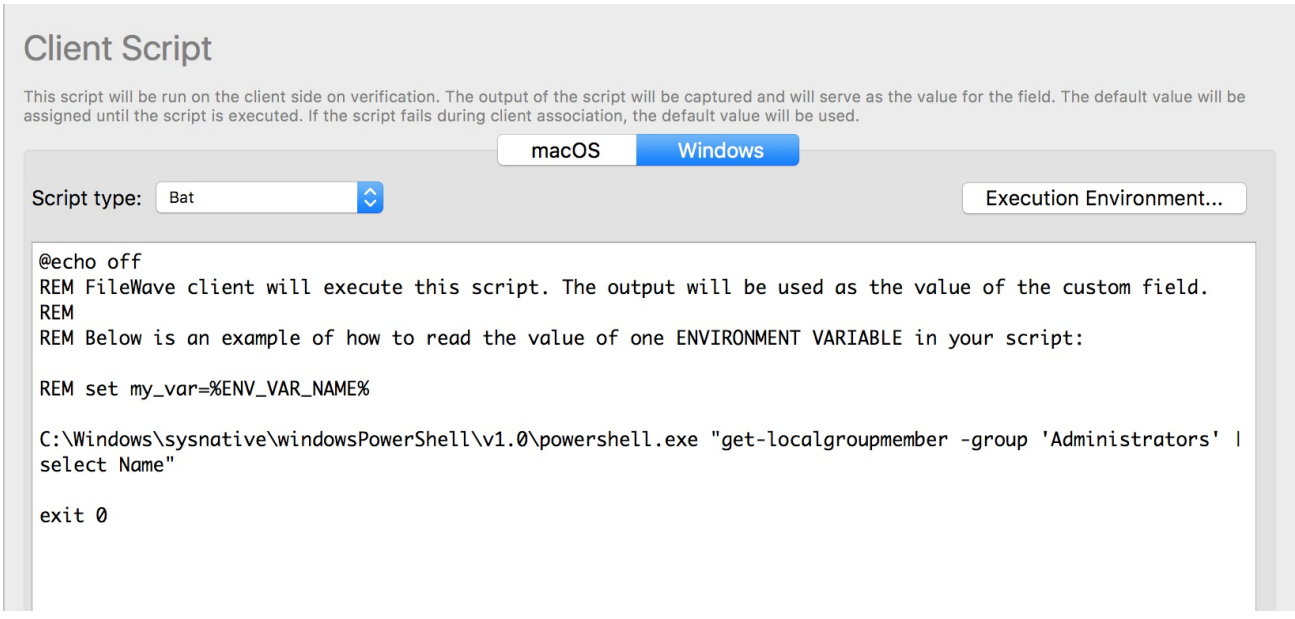
An example of this is with the below PowerShell command to list the local Admin accounts on a given device. The screen shot below shows how you would normally setup the Custom Field in your environment. The issue is that this command needs to be run in a 64 bit environment while FileWave defaults to 32 bit when executing the commands.

```
get-localgroupmember -group 'Administrators' | select Name
```



Since this is the case you will now need to modify the command choosing to run in 64 bit. To do this you would add "C:\Windows\system32\windowsPowerShell\v1.0\powershell.exe" to the command and run it as a Bat script instead of PowerShell in the Custom Field. I posted the code that would work with the Custom Fields below as well as a screen shot of the correct setup.

```
C:\Windows\system32\windowsPowerShell\v1.0\powershell.exe "get-localgroupmember -group 'Administrators' | select Name"
exit 0
```



Another method to execute an entire script in the native environment (32bit on 32bit or 64bit on 64bit) is as follows:

```
################################################################################
#If Powershell is running the 32-bit version on a 64-bit machine, we
#need to force powershell to run in 64-bit mode .
################################################################################
if ($env:PROCESSOR_ARCHITEW6432 -eq "AMD64") {
    #write-warning "Take me to 64-bit....."
    if ($myInvocation.Line) {
        &"$env:WINDIR\system32\windowspowershell\v1.0\powershell.exe" -NonInteractive -NoProfile
$myInvocation.Line
    }else{
        &"$env:WINDIR\system32\windowspowershell\v1.0\powershell.exe" -NonInteractive -NoProfile -file
"$($myInvocation.InvocationName)" $args
    }
exit $lastexitcode
}

# Main script
# Uncomment the next line to prove that we are always in 64bit
#[Environment]::Is64BitProcess

# Your 64bit script here.

################################################################################
#End
################################################################################
```

Related Content

- Scripting Languages supported in FileWave

Importing and Exporting Custom Field Files

Starting in FileWave version 13.1 you can import and export custom field definitions. This is ideal for sharing powerful fields not built into standard inventory.

> Always check a script before deploying to all devices. No, really, always!

Exporting:

1. Open your (Assistance → Custom Fields →) "Edit Custom Fields" UI
2. Select one or more custom fields
3. Press "Export"
4. Save the "FileWave Custom Fields.customfields" file
5. Share online, or store for later

Importing:

1. Download the custom field file
2. Open your (Assistance → Custom Fields →) "Edit Custom Fields" UI
3. Press "import" Browse for file
4. If the fileset contains a script, verify it is safe for your environment
5. Associate the field with the needed device(s)
6. You may also want to import custom field values

See Custom Fields for more on assigning fields to devices, and importing custom field values.

Conflicts:

Custom vs Internal

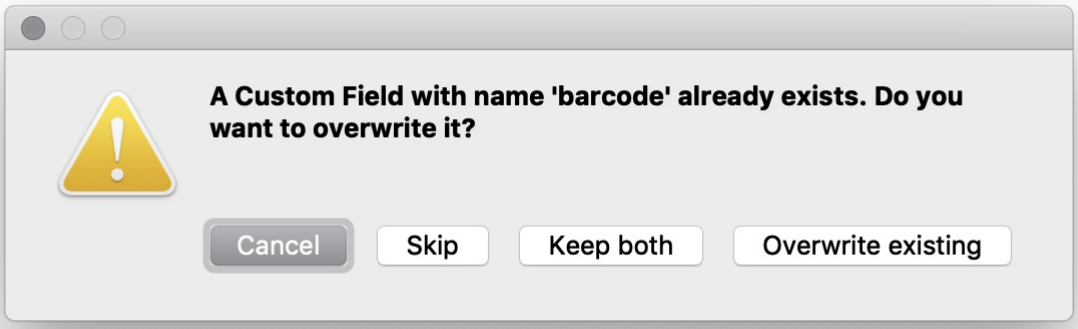
FileWave will let you re-use an internal name that is the same as another inventory element. If a Custom Field were added with an internal name of 'location', two 'location' internal names would co-exist the built-in internal name and the added Custom Field.

A prefix may be added to avoid confusion on matching inventory names. The prefix varies depending upon the method being used to reference the internal name.

Method	Prefix	Example
MDM Profile	custom_field	%custom_field.location%
Scripted (Launch Argument or Environment Variable)	CustomFields	%CustomFields.location%

Custom vs Custom

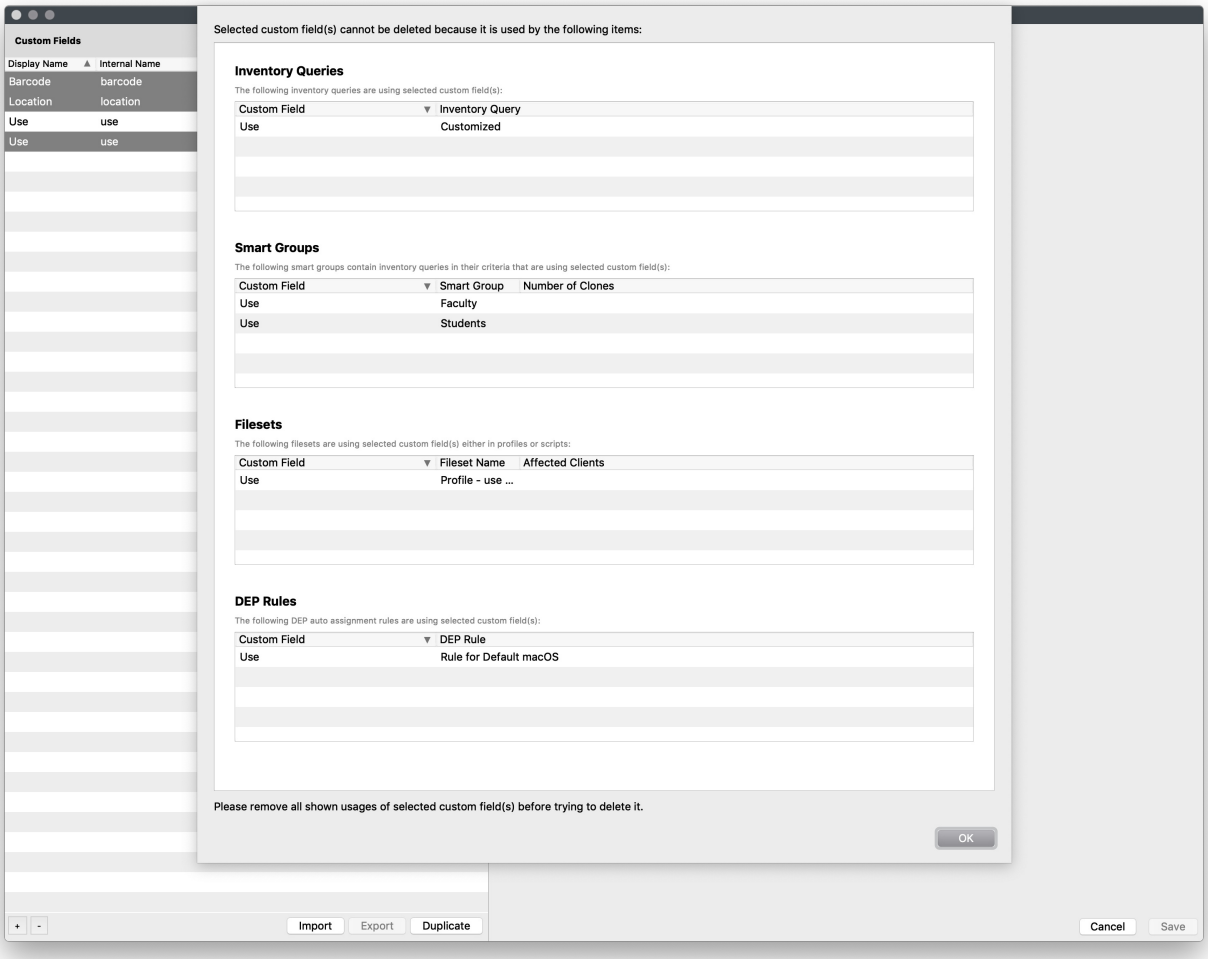
If you are importing a custom field with an internal name that is the same as an existing custom field internal name you will see:



- Cancel - Stops the whole process and makes no changes
- Skip - If importing a custom fields file with multiple entries you can skip the one that is conflicting and continue
- Keep both - This will add _2 to the internal name of the field you are importing
- Overwrite existing - Replace your existing entry entirely (as long as it isn't used in smart groups, inventory queries, etc)

Just because you CAN overwrite an existing custom field doesn't mean that is a good idea. Best practice would be to import it as an additional field and resolve the conflict manually.

If the internal names are in use (Smart Groups, Inventory Queries , DEP auto) when you press "overwrite existing" you will see this telling you where it is in use:



Custom Fields and Multi-Lined Outputs

Description

Custom Fields are an excellent way to provide additional reporting information on devices that are not included as standard inventory. However, there are alternative methods to present this information where the output includes large amounts of text.

Information

Scripted Custom Fields values are based upon the output of the included script. Although the information desired may be a list of items, sometimes it may be better to return an alternate response to that information and instead store the returned list of items differently. As such, the decision then becomes:

- What is the best information to return as a value, which can concisely provide meaningful information
- Where is the best place to store the required values for reference

As an example, consider 32-bit Apps on macOS 10.15 that are no longer supported. The list of responses from such a script could be vast, would be practically unreadable and make searching responses awkward. Instead, consider perhaps counting the amount of Apps stored on a device that are 32-bit and return this as an integer. Devices may then be targeted based upon the amount of 32-bit Apps that are installed.

The required list of 32-bit Apps, though, still needs to be stored, preferably located in an easy to obtain location for analysis.

Details

Continuing with the example, as of macOS 10.15 Apple dropped support for 32-bit Apps. Consideration was then required for listing 32-bit Apps on devices prior to upgrading to 10.15, such that software upgrades or alternate software could be implemented to replace those utilised. For this example the chosen options are:

- Return the quantity of 32-bit Apps as an Integer
- Output the list of 32-bit Apps into the FileWave Client log.

As an integer, Inventory Queries may be built to target the devices with the highest quantity of installed 32-bit Apps first and over time, as software is replaced, the count should come down, until all apps have been accounted for.

The FileWave Client log is easily accessible from the Admin console. Additionally, by tailoring the output with desired key words, observing this information then becomes easy

The script could look something like the following, which consists of:

- A python section to provide a dated format to match the current log
- A function to define the output if either an Application or Framework has been located (consideration has been made to identify Apps with Apps or within Frameworks, etc)
- A default output for any other identified binary, library, etc
- A command to identify the apps, pushed into a loop to trigger the above
- A total count to report back as the overall integer value

```
#!/bin/zsh

testline=""
linecount=0
logfile=/private/var/log/fwcld.log

function time_formatted {
python - <<END
import datetime

now = datetime.datetime.now()
year = '{:04d}'.format(now.year)
month = '{:02d}'.format(now.month)
day = '{:02d}'.format(now.day)
hour = format(now.hour)
minute = '{:02d}'.format(now.minute)
second = '{:02d}'.format(now.second)
mlsec = format(now.microsecond)
the_time = '{}-{}-{} {}:{}:{}.{}'.format(year, month, day, hour, minute, second, mlsec[:3])

print(the_time)
```

```
END
}

function printline {

        if [[ "$@" != "$testline" ]]
        then
                linecount=$(( linecount + 1 ))
                echo -e "\t$linecount: $@"  >> "$logfile"
        fi

        testline="$@"
}

echo $(time_formatted)"|main|INFO|CLIENT|Checking for 32bit apps" >> "$logfile"

while read line
do
        case "$line" in

                *".framework"*)
                        printline $(echo "$line" | awk -F "\\\.framework" '{print "32bit
app|Application|"$1".framework"}')
                        ;;
                *".app"*)
                        printline $(echo "$line" | awk -F "\\\.app" '{print "32bit app|Framework|"$1".app"}')
                        ;;
                *)
                        linecount=$(( linecount + 1 ))
                        echo -e "\t$linecount: 32bit app|Other|$line" >> "$logfile"
                        ;;
        esac
done< <(mdfind "kMDItemExecutableArchitectures == '*i386*' && kMDItemExecutableArchitectures != '*x86*'")

echo $linecount
echo -e "\tTotal 32bit apps found: $linecount" >> "$logfile"
```

Viewing the list output to the log file may be achieved via Client Monitor > Client Log and then by searching for '32bit app', which may look something like:

Pulling a log via client monitor requires a connection from your admin directly to the client. Client firewall, NAT, and Network ACL might prevent this from happening. Default TCP and UDP Port Usage

```
2019-07-20 15:46:31.373|main|INFO|CLIENT|Checking for 32bit apps
        1: 32bit app|Framework|/Users/rstehpens/Documents/Papercut MF Clent/mac/legacy/PCClient.app
        2: 32bit app|Framework|/Users/rstehpens/Documents/Papercut MF Clent/mac/legacy/client-local-install.app
        3: 32bit app|Framework|/Applications/BBEdit.app
        4: 32bit app|Framework|/Applications/Adobe Connect/Adobe Connect.app
        5: 32bit app|Framework|/Applications/RingCentral Meetings.app
        6: 32bit app|Application|/System/Library/Frameworks/CoreServices.framework
        7: 32bit app|Application|/System/Library/Frameworks/Carbon.framework
        8: 32bit app|Other|/usr/bin/vmmap32
        9: 32bit app|Other|/usr/bin/stringdups32
        10: 32bit app|Other|/usr/bin/malloc_history32
        11: 32bit app|Other|/usr/bin/leaks32
        12: 32bit app|Other|/usr/bin/heap32
        13: 32bit app|Other|/Users/rstehpens/Desktop/UserData/BigHonkingText
        14: 32bit app|Application|/System/Library/Frameworks/QuickLook.framework
        15: 32bit app|Framework|/System/Library/Input Methods/InkServer.app
        16: 32bit app|Application|/System/Library/Frameworks/DrawSprocket.framework
        17: 32bit app|Application|/System/Library/Frameworks/Scripting.framework
        18: 32bit app|Application|/System/Library/Frameworks/DVComponentGlue.framework
        19: 32bit app|Application|/System/Library/Frameworks/QuickTime.framework
        20: 32bit app|Application|/System/Library/Frameworks/AppKitScripting.framework
        21: 32bit app|Application|/System/Library/Frameworks/DiscRecording.framework
        22: 32bit app|Application|/System/Library/PrivateFrameworks/CoreMediaAuthoring.framework
        23: 32bit app|Application|/System/Library/PrivateFrameworks/vmutils.framework
        24: 32bit app|Application|/System/Library/PrivateFrameworks/CoreMediaPrivate.framework
        25: 32bit app|Application|/System/Library/PrivateFrameworks/CoreMediaIOServicesPrivate.framework
```

```
        26: 32bit app|Application|/System/Library/PrivateFrameworks/GraphicsAppSupport.framework
        27: 32bit app|Application|/System/Library/PrivateFrameworks/FWAVCPrivate.framework
        28: 32bit app|Other|/System/Library/Printers/Libraries/libConverter.dylib
        29: 32bit app|Other|/sbin/autodiskmount
        30: 32bit app|Other|/usr/sbin/pictd
        31: 32bit app|Other|/usr/lib/libnetsnmp.5.2.1.dylib
        32: 32bit app|Other|/usr/bin/qtdefaults
        33: 32bit app|Other|/usr/bin/qc2movie
        34: 32bit app|Framework|/usr/local/bin/iHook.app
        35: 32bit app|Other|/Users/rstehpens/Library/Android/sdk/build-tools/28.0.3/mipsel-linux-android-ld
        36: 32bit app|Other|/Users/rstehpens/Library/Android/sdk/build-tools/28.0.3/i686-linux-android-ld
        37: 32bit app|Other|/Users/rstehpens/Library/Android/sdk/build-tools/28.0.3/arm-linux-androideabi-ld
        38: 32bit app|Other|/Users/rstehpens/Library/Android/sdk/tools/mksdcard
        39: 32bit app|Framework|/Library/Application Support/Microsoft/Silverlight/OutOfBrowser/SLLauncher.app
        40: 32bit app|Other|/Users/rstehpens/Scripts/bin/BigHonkingText
        41: 32bit app|Other|/Users/rstehpens/Downloads/Safari/BigHonkingText106
        42: 32bit app|Framework|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/Meeting
Center.app
        43: 32bit app|Other|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/libopenh264-
0.6.6.0.dylib
        44: 32bit app|Other|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/libcrypto-
1.6.2.9.dylib
        45: 32bit app|Other|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/libssl-1.6.2.9.dylib
        46: 32bit app|Framework|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/atmsupload.app
        47: 32bit app|Framework|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/convertpdf.app
        48: 32bit app|Other|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/xml-31.0.0.1.dylib
        49: 32bit app|Other|/Users/rstehpens/Library/Application Support/WebEx Folder/T31_UMC/cmcrypto-
31.11.0.1.dylib
        50: 32bit app|Framework|/Users/rstehpens/Library/Application Support/WebEx
Folder/T31_UMC/asannotation2.app
        Total 32bit apps found: 50
```

As can be seen, this list would be problematic as a returned value in a Custom Field, but the combination of a single response and the easily obtainable list provides a neater solution.

There are other ways the information could be stored or viewed. Scripts ran through Filesets, e.g. Activation, Preflight, etc. store the output and may be viewed via Fileset Status. These scripts have the difference in that they are usually ran just once, which may be better or worse depending upon the use case. Additionally, scripts can include API calls to alter values in Administration Custom Fields.

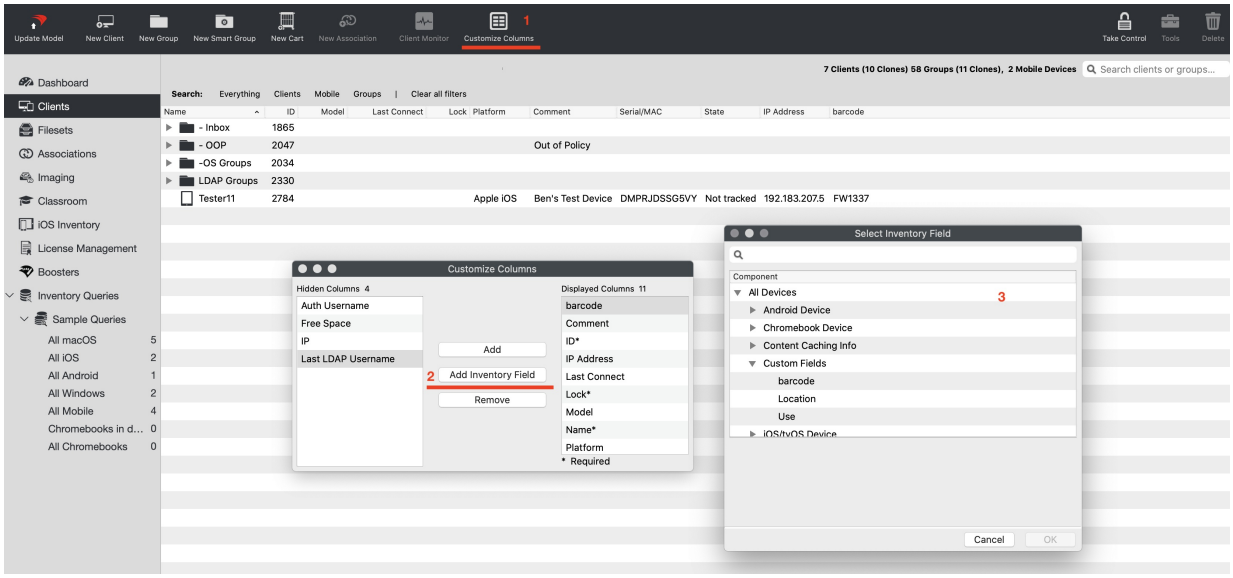
Each required report may have different styles of output, but with some consideration, reporting may be made easier with smarter application.

Custom Fields in Client View

Starting in FileWave13.3.0 you can add any inventory field to the client view.

Steps to add new fields

1. From the client view press the "Customize Columns" button
2. Press "Add Inventory Field" for advanced inventory options
3. Select any Inventory field (including custom fields)



> When you select an Inventory Field it is not hidden the next time you open the 'Select Inventory UI'
>
> Selecting a field that is already added will just close the UI, you can then open it again and select a field that is not already added.

> You will notice the fields are limited in the "Select Inventory Fields" UI as compared to the QueryBuilder UI. This is because we are only showing direct 1:1 inventory fields.

> Hiding the built-in "IP" column, and adding the Inventory: All Devices / "IP Address Column" does not change that IP client monitor uses.

Related Content

- See Custom Fields for more.

Add FileWave Custom Inventory fields remotely using a Fileset

Description

> Although these forms of Custom Field may still be used, since FileWave 12.7 additional new methods of Custom Fields have been introduced. The newer form allow for backward compatibility to these older Custom Fields, but provide much greater flexibility and would be recommended. Details on these may be found here: Custom Fields

Let's say you are using FileWave Integrated Inventory and you want to collect inventory data fields that FileWave does not report yet (you can check if you field is supported in the component list when you edit a query). The solution is to use the custom inventory fields introduced in FileWave version 7.0.0. There are 20 fields for each data type : String, Integer, Boolean, Date. Starting in version 8.5 the FileWave Client can accept input.

You can use this recipe to deploy a Fileset that is able to add the value you want based on a shell command. The script is already written for you and available in the Fileset, you only need to provide the command that outputs the value you want to add.

```
$ fwcld -custom_write -key <key_name> [-value <value_to_save] [-silent]
```

Ingredients

- FW Central
- A command that outputs needed infrmation
- Basic understanding of scripting

Directions

Built-into the FileWave Client is the ability to receive the output of a command and save that to the inventory DB.

1. Determine what the output of your command is and the best place to save

String	Integer	Boolean	DateTime
custom_string_01	custom_integer_01	custom_bool_01	custom_datetime_01
custom_string_02	custom_integer_02	custom_bool_02	custom_datetime_02
custom_string_03	custom_integer_03	custom_bool_03	custom_datetime_03
custom_string_04	custom_integer_04	custom_bool_04	custom_datetime_04
custom_string_05	custom_integer_05	custom_bool_05	custom_datetime_05
custom_string_06	custom_integer_06	custom_bool_06	custom_datetime_06
custom_string_07	custom_integer_07	custom_bool_07	custom_datetime_07
custom_string_08	custom_integer_08	custom_bool_08	custom_datetime_08
custom_string_09	custom_integer_09	custom_bool_09	custom_datetime_09
custom_string_10	custom_integer_10	custom_bool_10	custom_datetime_10
custom_string_11	custom_integer_11	custom_bool_11	custom_datetime_11
custom_string_12	custom_integer_12	custom_bool_12	custom_datetime_12
custom_string_13	custom_integer_13	custom_bool_13	custom_datetime_13
custom_string_14	custom_integer_14	custom_bool_14	custom_datetime_14
custom_string_15	custom_integer_15	custom_bool_15	custom_datetime_15
custom_string_16	custom_integer_16	custom_bool_16	custom_datetime_16
custom_string_17	custom_integer_17	custom_bool_17	custom_datetime_17
custom_string_18	custom_integer_18	custom_bool_18	custom_datetime_18
custom_string_19	custom_integer_19	custom_bool_19	custom_datetime_19
custom_string_20	custom_integer_20	custom_bool_20	custom_datetime_20

- Create a fileset with a script where it uses the clients

macOS

```
/usr/local/sbin/FileWave.app/Contents/MacOS/fwcld -custom_write -key FIELD_TO_SAVE_TO -value INFORMATION_TO_SAVE
```

Windows

```
C:\Program Files (x86)\FileWave\fwcld -custom_write -key FIELD_TO_SAVE_TO -value INFORMATION_TO_SAVE
```

- Associate this fileset

> Date format may be supplied as either YYYY/MM/DDTHH:MM:SS or YYYY-MM-DDTHH:MM:SS. E.g 2014/02/20T15:22:43 or 2014-02-20T15:22:43

Examples

Setting "custom_bool_13" to a false:

```
$ fwcld -custom_write -key custom_bool_13 -value 0
$ fwcld -custom_write -key custom_bool_13 -value false
```

Setting "custom_bool_13" to true:

```
$ fwcld -custom_write -key custom_bool_13 -value 1
$ fwcld -custom_write -key custom_bool_13 -value true
$ fwcld -custom_write -key custom_bool_13 -value something
```

Setting "custom_date_02" to a date:

```
$ fwcld -custom_write -key custom_date_02 -value 2014-02-20T15:22:43
```

To remove any key value, just leave off the -value parameter - so to reset the "custom_date_02" value back to it's default.

```
$ fwcld -custom_write -key custom_date_02
```

Example: Saving admins to string 01

```
#!/bin/sh
# This script is a verification sample
# benm @ fw

now=$(date +"%Y-%m-%d-%H-%M")
echo "$now -- Writing current admins to inventory"

#writes the current administrators to an inventory field
currentadmins=$(dscacheutil -q group -a name admin |grep users)

/usr/local/sbin/FileWave.app/Contents/MacOS/fwcld -custom_write -key custom_string_01 -value "$currentadmins"
```

Notes

If you set your script to run at the "verification" phase then it will continue to run (default every 24hrs), for more on scripts see: Fileset Scripts.

Useful Commands

- The current logged in user: `stat -f%Su /dev/console`
- The Kernel version: `uname -r`
- Battery Condition: `system_profiler SPPowerDataType | awk '/Condition/ {print $NF}'`
- Current admins: `dscl . read /Groups/admin GroupMembership | cut -d " " -f 2-`

Example Custom Fields

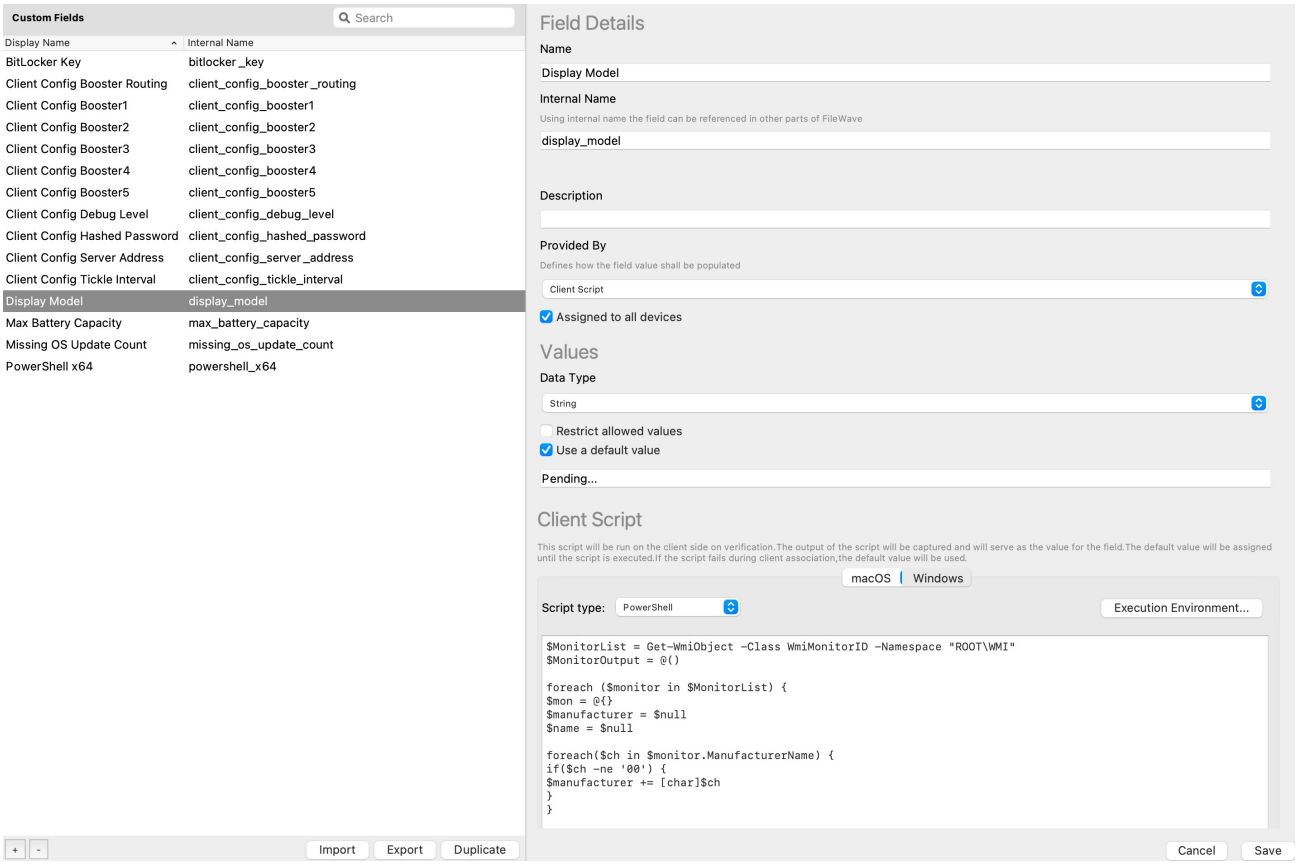
Some example fields.

Detecting the Display Model using a Custom Field

Use a FileWave Custom Field to store the model of display connected to a Mac or Windows system.

Adding the Custom Field

1. Download the following Custom Field export - Display Model Custom Field.customfields
2. Import the downloaded file into "FileWave Admin>Assistants>Custom Fields>Edit Custom Fields>Import".
3. Save changes within Custom Fields dialog.
4. Associate Custom Field with desired Windows devices via "right-click>Edit Custom Field(s) Associations" or check the "Assigned to all devices" checkbox when editing the Custom Field.
   - A Smart Group is very helpful to quickly associate Custom Field



- Copy/paste the following PowerShell script into the scripting box under the Windows section:

```
$MonitorList = Get-WmiObject -Class WmiMonitorID -Namespace "ROOT\WMI"
$MonitorOutput = @()

foreach ($monitor in $MonitorList) {
$mon = @{}
$manufacturer = $null
$name = $null

foreach($ch in $monitor.ManufacturerName) {
if($ch -ne '00') {
$manufacturer += [char]$ch
}
}

foreach($ch in $monitor.UserFriendlyName) {
if($ch -ne '00') {
$name += [char]$ch
}
```

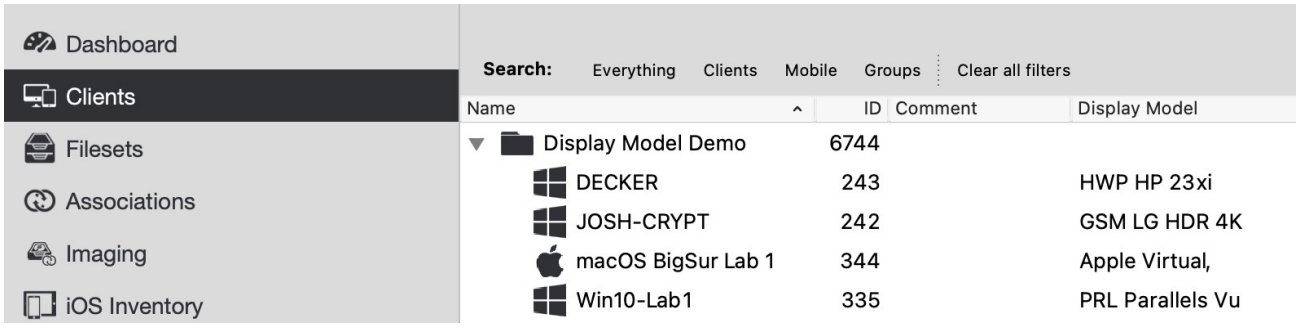
```
}
$mon = $manufacturer + " " + $name
$MonitorOutput += $mon
}
$MonitorOutput=$MonitorOutput -join ', '
$MonitorOutput
exit 0
```

- Copy/paste the following Shell script into the scripting box under the macOS section:

```
#!/bin/bash
DisplayModel=`system_profiler SPDisplaysDataType | grep "Resolution:" -B1 | awk -v n=3 'NR%n==1' | sed "s/^[
\t]*//" | sed 's/:/,/g' | tr '\n' ' '`
echo ${DisplayModel}
exit 0
```

- Save changes within Custom Fields dialog.
- Associate Custom Field with desired devices via "right-click>Edit Custom Field(s) Associations" or check the "Assigned to all devices" checkbox when editing the Custom Field.
  - A Smart Group can be helpful to quickly associate Custom Field.
- Alternatively you can also simply assign the field to all devices.

Results



Related articles

- Custom Fields