WORK-FLOW

Specialized Flow Cytometry Solutions

FlowJo Script Editor

1. Preface

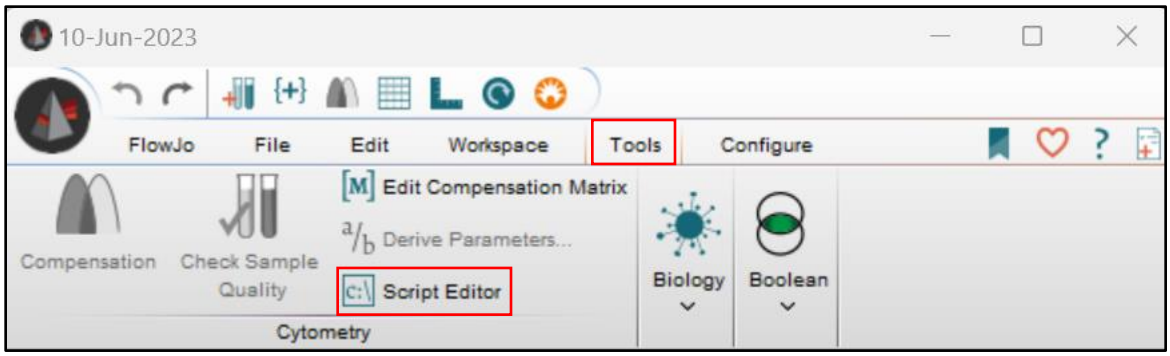
FlowJo Script Editor is a JavaScript platform that allows users to automate various operations such as generating keywords and executing mathematical calculations.

This guide explains how to use the Script Editor and demonstrates its utility in two use cases:

- CytoFLEX plate and well identification
- Flowrate and volume keyword extraction and cell concentration calculation

2. The Script Editor

- The Script Editor can be found under the Tools menu as shown below:



- You can start from scratch by typing directly in the Script Editor Panel or by loading a saved script (.fjs) via the Open Script button, then clicking the Run button. Additionally, you can perform calculations and test code excerpts by entering them into the Input Console and pressing “Enter”. In all cases, the results appear in the Output Console Panel.

WORK-FLOW
Specialized Flow Cytometry Solutions

3. Example 1 – CytoFLEX Well Identifier

The following script iterates over an array of CytoFLEX samples, extracting specific keywords (`'PLTNO'` and `'$FIL'`) from each FCS file and performing string manipulations to construct a complete well identifier. Finally, it assigns a new Well ID keyword to each file.

```
var samples = workspace.samples;
for (var i = 0; i < samples.length; i++) {
var currentsample = samples[i];
var plt = currentsample.keywords['PLTNO'];
var fil = currentsample.keywords['$FIL'];
var withoutExt = fil.substring(0, fil.lastIndexOf('.'));
var wellid = 'Plate' + plt + '_' + withoutExt.substring(withoutExt.lastIndexOf('-') + 1, withoutExt.length);
currentsample.keywords['Well ID'] = wellid;}
```

Results

Two samples are loaded in the Workspace:

Name	Statistic	#Cells
01-Well-A1.fcs		10000
02-well-E1.fcs		2679

The script is run and a new Well ID column is automatically created:

Name	Statistic	#Cells	Well ID
01-Well-A1.fcs		10000	Plate01_A1
02-well-E1.fcs		2679	Plate02_E1

Here is the breakdown of the code:

`var samples = workspace.samples;`

This line initializes a variable named `samples` and assigns it the value of `workspace.samples`. It suggests that `workspace` is an object or context that contains a property called `samples`.

`for (var i = 0; i < samples.length; i++) {`

This is a “for loop” that iterates over the `samples` array. It uses the variable `i` as the loop counter, starting from 0 and incrementing by 1 until it reaches `samples.length`.

`var currentsample = samples[i];`

Within each iteration of the loop, this line assigns the current element of the `samples` array to the variable `currentsample`. It allows the code to access and manipulate the properties of the current sample.

`var plt = currentsample.keywords['PLTNO'];`
This line extracts the value of the `'PLTNO'` property from the `keywords` object of the current sample and assigns it to the variable `plt`.

`var fil = currentsample.keywords['$FIL'];`
Similarly, this line retrieves the value of the `'FIL'` property from the `keywords` object of the current sample and assigns it to the variable `fil`.

`var withoutExt = fil.substring(0, fil.lastIndexOf('.'));`
Here, the code extracts the substring of `fil` starting from index 0 up to the last occurrence of the period ('.'), effectively removing the file extension (fcs). The resulting substring is stored in the `withoutExt` variable.

`var wellid = 'Plate' + plt + '_' + withoutExt.substring(withoutExt.lastIndexOf('-') + 1, withoutExt.length);`
This line constructs the `wellid` string by concatenating the following parts:
- `'Plate'`: A static string indicating the plate.
- `plt`: The extracted plate number from the `'PLTNO'` property.
- `'_'`: A static underscore character.
- `withoutExt.substring(withoutExt.lastIndexOf('-') + 1, withoutExt.length)`: A substring of `withoutExt`, starting from the character immediately following the last occurrence of a hyphen ('-') and extending until the end of the string. This substring likely represents a well identifier.

`currentsample.keywords['Well ID'] = wellid;`
Finally, this line assigns the `wellid` value to the `'Well ID'` property of the `keywords` object within the current sample. It adds or updates the `'Well ID'` property with the calculated well identifier.

4. Example 2 – Flowrate, Volume and Concentration

The following script retrieves the #FLOWRATE (if available), $VOL (sample volume) and $TOT (total number of events in the data set) keywords from each FCS file then calculate the sample concentration using the following formula:

$TOT (cells) / $VOL (nL) * 1000 (nL/µL) = cells / µL

Note: FCS 3.1 contains the optional keyword $VOL or sample volume expressed in nanoliters. However, Cytek's $VOL keyword is expressed in microliters, making the 1000 factor unnecessary.

```
var samples = workspace.samples;
for (var i = 0; i < samples.length; i++) {
var currentsample = samples[i];
var flowrate = currentsample.keywords['#FLOWRATE'];
currentsample.keywords['Flowrate'] = flowrate;
var volume = currentsample.keywords['$VOL'];
currentsample.keywords['Volume'] = volume;
var concentration = currentsample.keywords['$TOT']/currentsample.keywords['$VOL'] * 1000;
currentsample.keywords['Concentration'] = concentration;}
```

Results

Two samples are loaded in the Workspace:

Name	Statistic	#Cells
CytoFLEX.fcs		10000
Attune.fcs		23842

The script is run and new columns are automatically created:

Name	Statistic	#Cells	Flowrate	Volume	Concentration
CytoFLEX.fcs		10000	org.mozilla.j...	48366.24	206.755786...
Attune.fcs		23842	200	19000	1254.84210...

Note: The Attune NxT adds a flow rate keyword expressed in µL/min but the CytoFLEX does not, which explains the “org.mozilla.javascript.undefined@1ebf916b” error.