FlashIDA usage guide

General description

FlashIDA runs from a command line. When active it connects to the instrument and controls the acquisition of mass spectrometry data. The currently running MS method (if any) will be suppressed by the FlashIDA's one. The acquisition parameters are

	Display a short usage information and exit
`-v, --version`	Display version number and exit
`-o, --nocc`	Normally, the software will wait for the contact closure signal from the instrument before starting method execution, using this option one can override the contact closure to start the method immediately.
`-t, --test`	Run the software in the test mode. Do not connect to the instrument and read the spectral information from the text file. This option is only used for internal testing and, thus, won't be discussed in detail.
`-m, --method=VALUE`	Location of the method file, the value can be absolute or relative (in respect to the executable) path. If no value is specified "method.xml" file in the executable folder will be used.
`-r, --rawname=VALUE`	By default, the log files are created as "FlashLog_{TIMESTAMP}" and "IDALog_{TIMESTAMP}". This parameter allows customizing created log files for easier analysis afterward. The value can be a text (NAMEID)or a file path, in the latter case only the last part of it without the extension will be used as NAMEID. The log files will be named as "FlashLog_{NAMEID}" and "IDALog_{NAMEID}"

Structure of the method file

Method file contains parameters of MS1 and MS2 scans acquired by the instrument and FlashIDA parameters. It is a plain XML file and can be modified by the user, if necessary, please, note, however, that the XML element names and allowed values are case sensitive and must match exactly to the backend representation. The software comes with the exemplary method file – "method.xml". Below, is the description of all allowed XML elements.

Method parameters

These parameters are relevant for the method as a whole

Name	Description

TopN	The maximal number of fragment scans that will be scheduled for each MS1 scan, should be a positive counting number
Duration	The length of the method in minutes should be a positive b eg

RFLens	MS1	RF Lens value in %, should be a positive real number
SourceCID	MS1	In-source CID fragmentation energy in eV, should be a positive real number
IsolationMode	MS2	Select the device used for precursor isolation, allowed values are **Quadrupole** and **IonTrap**
Activation	MS2	Activation/Fragmentation method, the following methods are supported in the current version **CID**, **HCD**, **ETD.** EThcD scans can be made selecting ETD and non-zero Collision Energy (see below), the latter will be used as the value for the supplemental activation
CollisionEnergy	MS2	Normalized collision energy both for HCD and CID activation, should be positive counting number
ReactionTime	MS2	Reaction time for ETD fragmentation, for other fragmentation methods any value will be ignored, should be a positive real number
ReagentMaxIT	MS2	Maximum injection time for reagent ion in ETD, for other fragmentation methods any value will be ignored, should be a positive real number
ReagentAGCTarget	MS2	AGC target for reagent ion in ETD, for other fragmentation methods any value will be ignored, should be positive counting number compatible with the instrument

FlashIDA parameters

These parameters are relevant for the FlashIDA algorithm

Name	Description
MaxMs2CountPerMs1	, should be a positive counting number
QScoreThreshold	The quality score threshold for precursors to be considered for fragmentation should be a real number
MinCharge	The minimal charge to consider for spectral deconvolution should be a positive countable number
MaxCharge	The maximal charge to consider for spectral deconvolution should be a positive countable number
MinMass	The minimal precursor mass to consider for spectral deconvolution should be a positive real number

MaxMass	Maximal precursor mass to consider for spectral deconvolution should be a positive real number
Tolerances	Tolerance for mass deconvolution in ppm should be an array of two real numbers – upper and lower tolerances, represented as two **<double>** elements in XML
RTWindow	The expected retention time window (in seconds) for elution of the same proteoform, should be a positive real number

Changing logging

FlashIDA uses the **log4net** library to create logs. The configuration used by **log4net** is saved in the corresponding section of the **.config** file. This section will not cover the complete configuration of **log4net** logging, please, refer to the log4net manual for it.

Logging messages generated by the software are forwarded into three different places (appenders in log4net terms) – the console window, and two text files. It is relatively easy to change how many details are logged in each of the appenders, by changing **<threshold value>** element of an appender. For example, the following change

Before

```
<appender name="ConsoleAppender" type="log4net.Appender.ColoredConsoleAppender" >
…
  <threshold value="INFO" />
…
</appender>
```

After

```
<appender name="ConsoleAppender" type="log4net.Appender.ColoredConsoleAppender" >
…
  <threshold value="WARN" />
…
</appender>
```

will limit console output only to Warning and Error messages, Debug and Info messages will be still written to other appenders.

The general level of logging can be modified by changing **<level value>** existing **<logger>**s. Two loggers are used in FlashIDA – "**General**" for general info, and "**IDA**" for messages related to FlashIDA core. By default, both report all messages – i.e. Debug and higher level. The following example shows, how to exclude debug messages from logging

Before

```
<logger name="General">
  <level value="DEBUG" />
…
</logger>
<logger name="IDA">
```

```
  <level value="DEBUG" />
…
</logger>
```

After

```
<logger name="General">
  <level value="INFO" />
…
</logger>
<logger name="IDA">
  <level value="INFO" />
…
</logger>
```

Logging can be suppressed completely or in part, by disabling corresponding appenders in logger configuration. For example, the following change (marked lines) will disable text file logging, and will leave only console one.

```
<logger name="General">
  <level value="DEBUG" />
  <appender-ref ref="ConsoleAppender" />
  <!-- <appender-ref ref="GeneralFile" /> -->
</logger>
<logger name="IDA">
  <level value="DEBUG" />
  <appender-ref ref="IDAFile" />
  <!-- <appender-ref ref="IDAInfoForward" /> -->
</logger>
```

Running FlashIDA acquisition

FlashIDA controls only the MS part of the acquisition. It can be used on its own, for example, during direct infusion experiment, or “on top” of the existing LC-MS method, in the latter case, it will override any existing MS method. A single instance of FlashIDA has to be executed for each LC-MS run, although it is possible (no hardware damage should occur) to run multiple instances, the result will be unpredictable, since different instances, will schedule their task to the instrument concurrently. FlashIDA will exit and return the control to Tune or Xcalibur after the FlashIDA method has finished. Please, note, that in the very beginning of the FlashIDA method, one might expect, so-to-say, transition period, when the instrument will execute scans both from the Xcalibur method and FlashIDA, this happens due to the way the instrument work, and should not last longer than few seconds.

Setting up FlashIDA-only experiment

This type of experiment is relevant for the cases when you do not need/want to use any LC in front of the instrument, for example when performing the direct injection, or you want to run FlashIDA manually for some time.

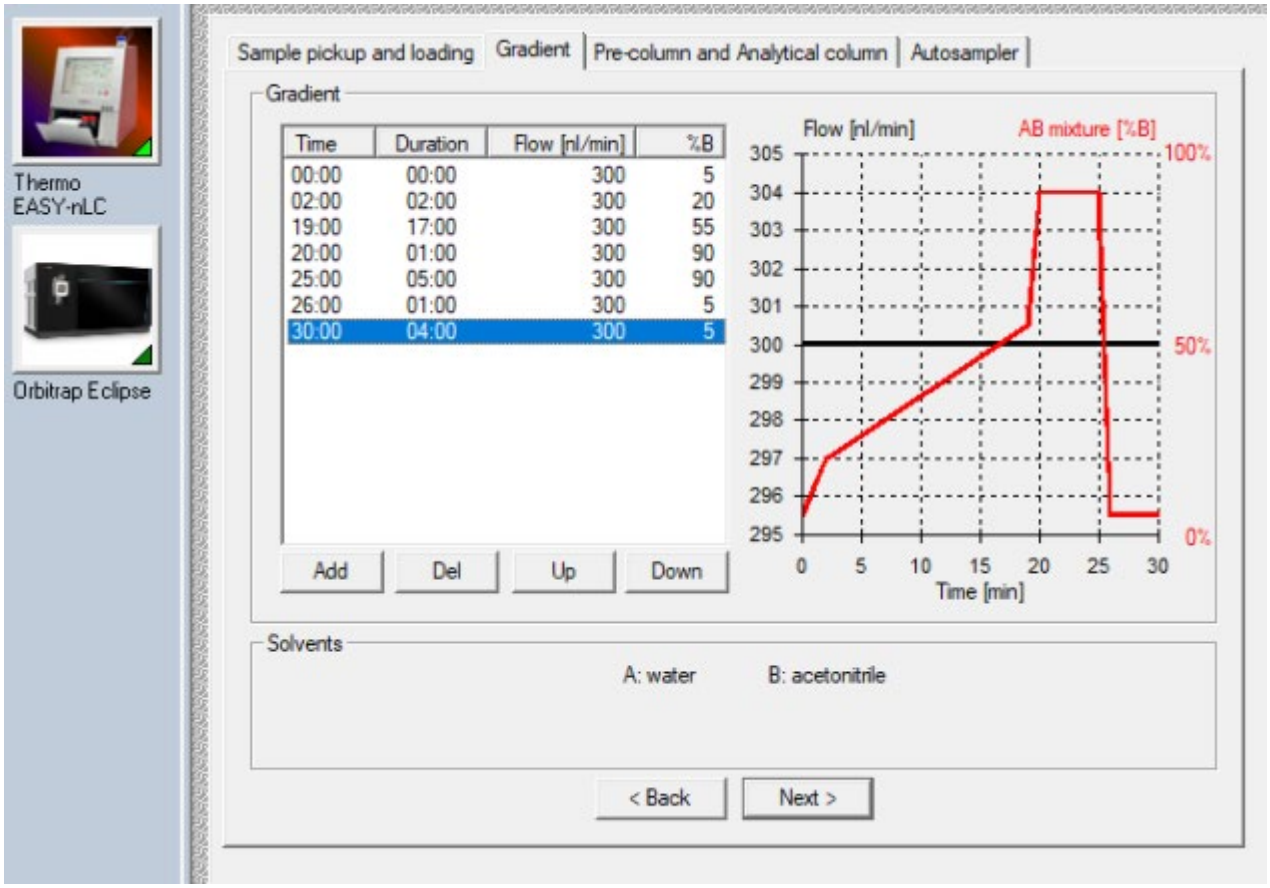
Make sure that all other parts of the experiment are running fine, for example, you have a stable spray, the molecule of interest is entering the mass spectrometer, and start FlashIDA executable with `-o, --nocc` command line parameter to start the acquisition ASAP. You will need to make sure that either **method.xml** file is appropriate to the experiment you want to execute, or provide `-m, --method` parameter pointing to

the location of the method file. If you do not have the acquisition running already, you will need to start it manually.

Setting up FlashIDA experiment through Xcalibur

This way of operation is handy when you want to use LC in front of the instrument, or want to acquire several samples in a sequence. Acquisition of the raw files is handled by Xcalibur, while FlashIDA is used to substitute the MS part of the experiment. FlashIDA can monitor the contact closure signal from the instrument and start the method using this signal.

1. Start setting up your LC-MS experiment as usual, i.e. create the LC part relevant for your needs. If your LC is not controlled by the Xcalibur software, you still can set up the MS-only method and use contact closure to start data acquisition.

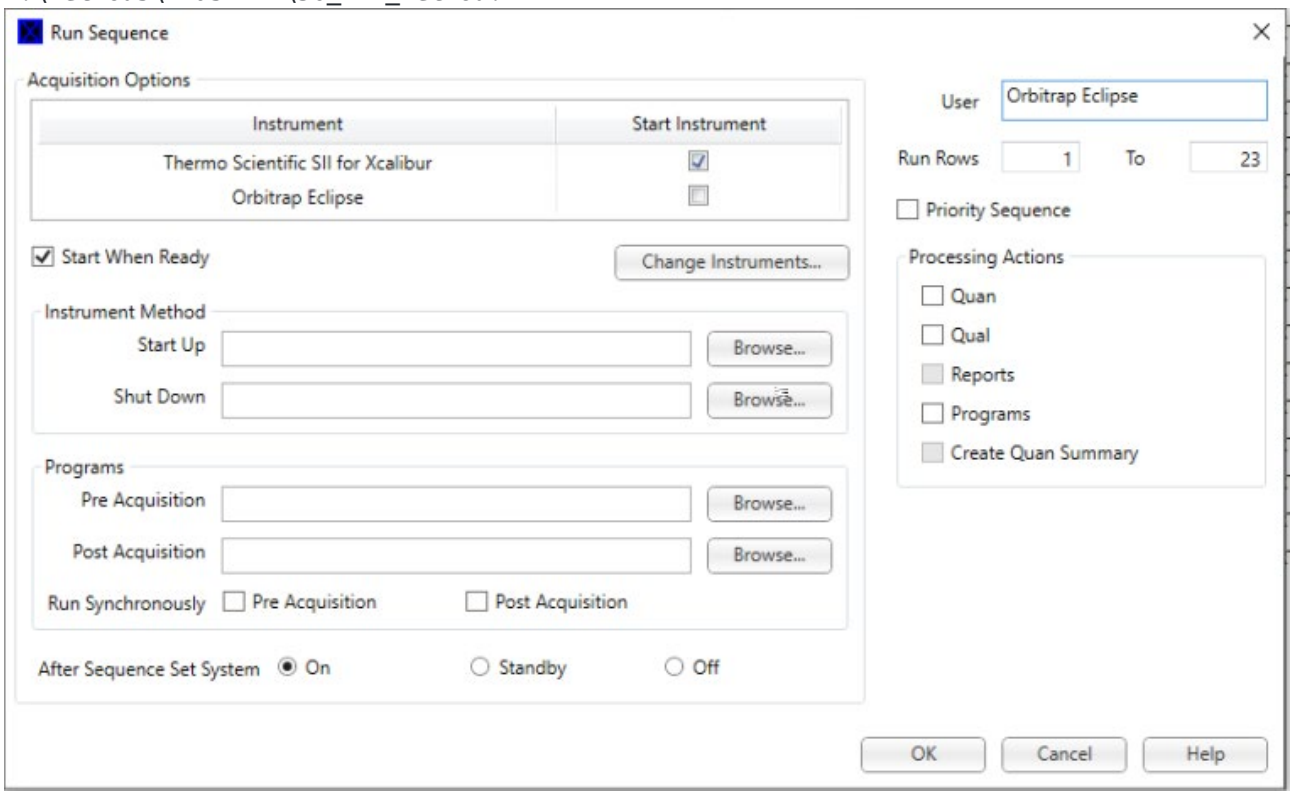


2. Make sure that the **Global Parameters** of MS method such as spray voltage, gas flows in the source, are set to the necessary values. There are not special requirements for the **Scan Parameters** part of the method since all settings here will be overridden by FlashIDA. It is advised, however, to set the total

duration of the method to match the LC method duration.

3. Adapt the existing **method.xml** file to your needs or create the new one with the settings of your experiment. Make sure that the duration of experiment in the method file does not significantly

```
D:\methods\FlashIDA\30_min_method.xml
```



5. Xcalibur allows sending some metadata of the running sample to command line tools. In particular, *%R: Provides the current raw data file*, this can be use together with the `-r, --rawname` parameter of FlashIDA to name the log-files according the raw file name. For example,

```
D:\FlashIDA\Flash.exe -m D:\methods\FlashIDA\30_min_method.xml -r %R
```

If any of the log-files exist, timestamp will be added to the filename. By default, log-files will use just a timestamp. For the complete list of available metadata refer to **Run Sequence** section of Xcalibur help.

Understanding log-files

FlashIDA uses two log files per each run, **FlashLog_** one is used for most of the information, while, **IDALog_** one only for the information from FlashIDA core, i.e. the information related to the targets found in each scan. By default, both logs are set for **Debug**-level output, the section above explains how to change the debugging level or disable it, if desired.

FlashIDA works in a loop, it receives MS scans from the instrument, analyze them, and form additional scan requests to be executed by the instrument. The loop starts with the start of the method and lasts until the duration of the method. FlashIDA can issue AGC, MS1, and MS2 scans requests. Each scan request produced by FlashIDA gets a unique numeric **Access ID** or just **ID** starting at 41. This identifier can be used to monitor the progression of each scan request. **Access ID** will be written into trailer information structure in the raw file and can be inspected by looking at scan header. All scan requests are stored in FIFO queue and submitted one after another to the instrument. Scan requests are then executed by the instrument; however, it is **not** guaranteed that the instrument will execute the scans requests in the same order as they are submitted.

FlashLog will report on which MS scans are received and sent to the instrument and the state of the FIFO queue. The number in front shows a timestamp (number of ms from the starting of the application). Below are the examples of log entries with explanation.

```
786766 Sending Full Orbitrap scan [500 - 2000]; ID: 44
```

A request to perform **Full MS1 Orbitrap** scan with mass range **500 – 2000** and Access ID **44** was sent to the instrument

```
787395 RECD FTMS MS1 Scan #8; ID: 44
```

The request was executed (Access ID **44)** and MS1 scan with scan number **8** was received by the software

```
787399 MS1 Scan# 8 RT 0.0317269848 (Aceess ID 44) - 0 targets
```

The same scan has been analyzed and **0** precursor targets were generated. Scan number and retention time match the ones written to the raw file.

```
1612381 ADD m/z 739.2156/2.29 (11+) qScore: 0.9101 to Queue as #4
```

Fragmentation scan for the target(precursor) m/z **739.2156**, with isolation window **2.29** Th, was added to the FlashIDA scan request queue at position **4**. The precursor has charge **11** and quality score **0.9101**

```
1612381 QUEUE is [#0 MSn Orbitrap [729.736236572266, 22+] - #4369 Full IonTrap [5
00:2000] - #4370 Full Orbitrap [500:2000] - #0 MSn Orbitrap [739.215637207031, 11
+] - #0 MSn Orbitrap [1445.72064208984, 9+] - #0 MSn Orbitrap [891.063781738281,
18+] - #0 MSn Orbitrap [757.001708984375, 17+] - #4369 Full IonTrap [500:2000] -
#4370 Full Orbitrap [500:2000]]
```

The condition of the scan request queue, each scan is represented by the short summary with type of the scan, analyzer, precursor m/z and charge (for MSn scans), and mass range for MS1 scans. Scan added in the previous example can found in the queue at position 4.

```
1612802 POP MSn scan MZ = 739.215637207031 Z = 11 // AGC: 1, MS1: 1, MS2: 3
```

The same precursor scan request is popped from the queue, the queue contains still **1** AGC scan, **1** MS1 scan, and **3** MS2 scans

```
1612803 Sending MSn scan MZ = 739.215637207031 Z = 11; ID: 4377
```

The fragmentation scan request was sent to the instrument and received ID **4377**

IDALog will report on the result of the analysis of the scans. See examples and explanations below.

```
MS1 Scan# 3650 RT 10.2560682368 (Aceess ID 3686) - 4 targets
```

MS1 scan with scan number **3650**, retention time **10.2560682368** (both match to raw file information), having Access ID **3686** was processed and **4** possible precursor targets were found. Each found precursor will be represented by the following text.

```
Mass=9969.86636706996        Z=11        Score=0.819953358681964 Window=[906.9416
87011719-908.870568847656]   PrecursorIntensity=1656068.5        PrecursorMassInt
ensity=3923630.359375        Features=[0.954601168632507,5.42687654495239,0.95818
1262016296,1.55814671516418,0.820968925952911,1.47546184062958]  ChargeRange=[11-
14]
```

It provides, targeted mass, charge, quality score, isolation window, as well as a number of other parameters, much of them are only useful for the debugging of the deconvolution process.