mzIdentML Validator Tutorial

**Salvador Martínez de Bartolomé, June 2013**

**Gerhard Mayer, February 2015**

# Introduction

The mzIdentML validator is a Java tool for the validation of mzIdentML files, checking whether their structure and semantic content follows the mzIdentML specification (<http://code.google.com/p/psi-pi/downloads/list>), as well as checking the [MIAPE](http://www.psidev.info/miape) compliance against the [MIAPE Mass Spectrometry Informatics](http://www.psidev.info/sites/default/files/MIAPE_MSI_1.1.pdf) module.

The mzIdentML validator is built as an extension of the PSI semantic validator framework ([reference](http://www.ncbi.nlm.nih.gov/pubmed/19834897)), a Java library that uses semantic rules, defined in an XML file, for checking that the appropriate controlled vocabularies are used in the correct positions in the standard xml files.

# Requirements

Java JRE 7.0 version or later is required. It can be installed from [here](http://java.com/en/download/index.jsp).

# How to get the mzIdentML valdator

The mzIdentML validator is available from the psi-pi google code page, where the code can be downloaded. The latest version of the stand-alone version of the validator is available at the download section: <https://code.google.com/p/psi-pi/downloads/list>. There is also a [Java Web Start](http://es.wikipedia.org/wiki/Java_Web_Start) version of the validator at: <https://psi-pi.googlecode.com/svn/trunk/validator/trunk/mzid-validator.html>.

# How to start

**The stand-alone version**

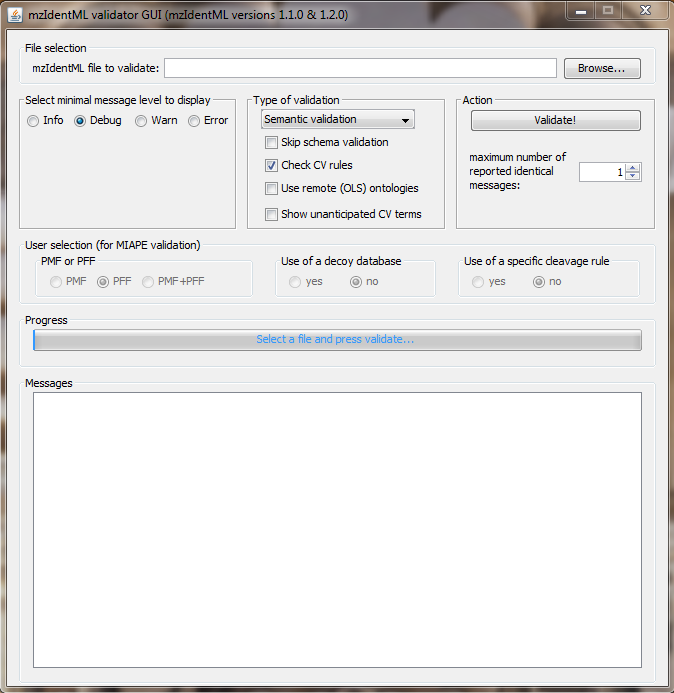
Download the latest version of the validator as commented before, as a zip file. Then decompress the file in a folder and start the “run.bat” batch file (windows) or the “run.sh” file (linux and mac).

**The web start version**

Click on the link [here](https://psi-pi.googlecode.com/svn/trunk/validator/trunk/mzid-validator.html) and press the button “Launch”. Then, the mzIdentML validator will start automatically after its loading. If not, open the “.jnlp” file with the “Java™ Web Start Launcher” available in your system.

# How to use it

After open the validator, an easy-to-use GUI will appear as:



Then, follow the next steps:

1. **Select the file**

Choose the mzIdentML to validate by clicking on “Browse” button.

1. **Select the message level to display**

The rules that are checked can have several severities: MAY, SHOULD or MUST. Depending on the level of message to display, the resulting error validation messages will be shown or not. Typically, we would recommend users to select “Warn”.

* If “Error” level is selected only fatal errors will be shown i.e. those that result in an invalid file.
* If “Warn” level is selected, errors that are highly recommended to get fixed are shown in addition to fatal errors.
* If “Info” level is selected, potential annotation improvements are shown in addition to warnings and fatal errors.

*For advanced users:*

* *If “Error” level is selected, just the messages from the failed rules with severity “MUST” will be shown.*
* *If “Warn” level is selected, the messages from the failed rules with severity “SHOULD” and “MUST” will be shown.*
* *If “Info” level is selected, all the messages from the failed rules with severity “MAY”, “SHOULD” and “MUST” will be shown.*

1. **Select the type of the validation**

* MIAPE compliant validation: The validator will check if the mzIdentML file is compliant with the MIAPE MSI guidelines defined by the HUPO-PSI (this is a higher level of validation beyond basic checking that the file is in the correct format).
* Semantic validation: The validator will check if the mzIdentML file is well formed and follows the official specifications.

Note that the MIAPE compliant validation already checks all the semantic rules, so if an mzIdentML passes the MIAPE validation, it is already semantically valid.

1. **Skip schema validation or not**

For large size mzIdentML files, the schema validation can take a while, so it is possible to skip that step in the validation process. This setting should be used with caution, since files not conforming to the XML schema may not be flagged up as invalid.

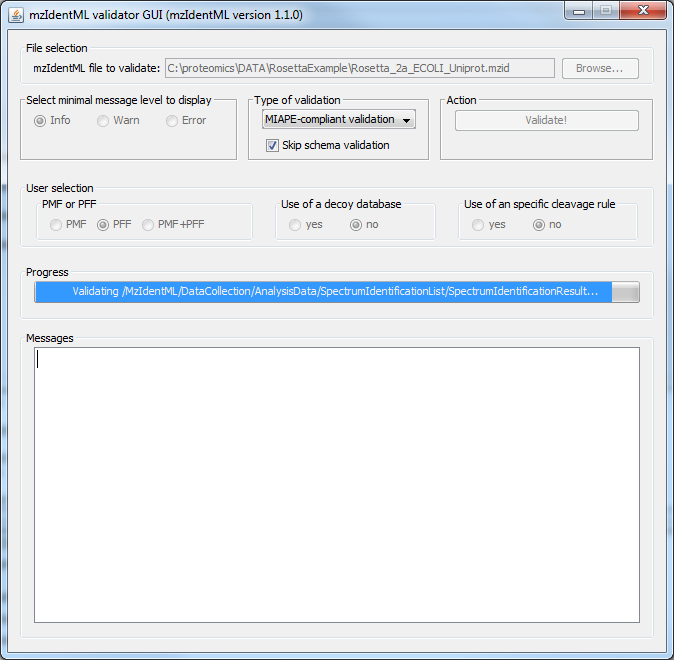
1. **Select the appropriate option in the user selections**

The user should select the appropriate options in the “user selection” panel of the validator. Depending on the selection, some rules that require additional annotations will be applied or skipped.

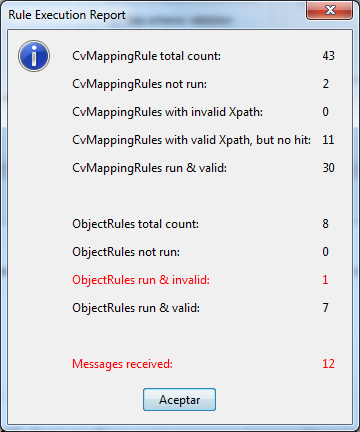
* *PMF or PFF*: select whether Peptide Mass Fingerprint (PMF) data or Peptide Fragment Fingerprint (PFF) data (i.e. tandem MS) was used to get the search results contained in the mzIdentML.
* *Use of a decoy database*: indicate whether a decoy database was used in the search or not.
* *Use of a specific cleavage rule*: indicate whether a specific/non-standard cleavage rule was used as input parameter in the search.

1. **Click on “Validate!” button and get validation messages**

Click on the “Validate!” button to start the validation. While the validation process is running, the progress bar will show the different steps occurring.



After the validation process is finished, a validation report is shown as a pop-up, displaying a summary of the rules that have been checked and the number of errors found:



After closing that dialog, the “Messages” text area of the tool will contain the set of validation messages from the rules that detected some errors in the file. For example:

Message 8:

Level: ERROR

Context(/MzIdentML/Provider )

--> There is not an affiliation name for the organizations referenced by the provider (id='PROVIDER') at /MzIdentML/Provider

Tip: Add a valid contact email ('MS:1000589') in any referenced element in the provider at /MzIdentML/Provider

Tip: Add an affiliation name ('MS:1000590') in any referenced element in the provider at /MzIdentML/Provider

That message indicates that the mzIdentML has an error in the /MzIdentML/Provider element of the file, in which an affiliation name MUST be present. Some tips to solve the problem are also shown.

# For advanced users

Some advanced users can validate their own mzIdentML files using their own rules, by modifying the following files in the application folder:

* **mzIdentML-mapping\_1.1.0.xml**: the cvMapping rules for the semantic validation. If this file is modified, the semantic validation will change according to the modifications in the rules. For more documentation about how to build your own cvMapping rules go to [here](http://www.psidev.info/validator-tutorial-0).
* **miape-msi-rules.xml**: the cvMapping rules for the MIAPE compliant validation. If this file is modified, the MIAPE compliant validation will change according to the modifications in the rules.
* **ruleFilter\_MIAPEMSI.xml** (see schema definition [here](http://proteo.cnb.csic.es/miape-api/schemas/ruleFilter_v1.3.xsd)): this file is used (just in the MIAPE validation) to define:
  + Some dependences between the options that the user selects in the GUI: depending on the user selection, some rules will be ignored or not. The options and the corresponding answers are:
    - USER\_SPECIFIC\_CLEAVAGE\_RULE:
      * USER\_SPECIFIC\_CLEAVAGE\_RULE or
      * NO\_USER\_SPECIFIC\_CLEAVAGE\_RULE
    - DATABASE\_TYPE
      * DECOY\_DATABASE
      * NO\_DECOY\_DATABASE
    - MASS\_SPECTRA\_TYPE
      * PMF
      * PFF
      * PMFPFF
  + Some dependences between rules: depending on the result of a certain rule, some others will be ignored or not.
    - This feature is not implemented in the current version of the validator.
    - It can be enabled by adding a “ruleConditions” element just after the “userConditions” element.
  + Some elements in the mzIdentML that MUST be present. If not, an ERROR level message will be shown.