Article

Error-free data visualisation and processing through imzML and mzML validation

Alan Mark Race, and Andreas Römpp

Anal. Chem., **Just Accepted Manuscript** • DOI: 10.1021/acs.analchem.8b03059 • Publication Date (Web): 17 Sep 2018

Error-free data visualisation and processing through imzML and mzML validation

Alan M. Race* and Andreas Römpp*
Universität Bayreuth, Bayreuth, Germany
E-mail: alan.race@uni-bayreuth.de; andreas.roempp@uni-bayreuth.de

Abstract

Open data formats are key to facilitating data processing, sharing and integration. The imzML format (http://imzml.org/) has drastically improved these aspects of mass spectrometry imaging data. Efficient processing of data depends on data sets which are consistent and adhere to the specifications, however this is not always the case. Here we present a validation tool for data stored in both imzML and the HUPO-PSI mass spectrometery counterpart, mzML, to identify any deviations from the published (i)mzML standard which could cause issues for the user when visualising or processing data. The tool is released in two forms, a graphical user interface (GUI) for ease of use, and a command line version to fit into existing workflows and pipelines. When certain known issues are encountered, such as the presence of negative values for the location of the binary data , the validator resolves the issue automatically upon saving. The GUI version of the validator also allows editing of the metadata included within the (i)mzML files in order to resolve inconsistencies. We also present a means of performing conditional validation on the metadata within (i)mzML files, where user-defined rules are validated against depending on whether specific metadata are present (or not). For example, if the MALDI term is present, then additional rules related to MALDI (such as the requirement of inclusion of laser parameters) can be validated against. This enables a flexible and more thorough automated validation of (i)mzML data. Such a system is necessary for validating against more comprehensive sets of metadata such as minimum reporting guidelines or metadata requirements prior to submission and acceptance of data to data repositories. We demonstrate how this tool can be used to validate against the proposed minimum reporting guidelines in MSI as well as institute specific metadata criteria. The validator tool is endorsed for validation of imzML (http://imzml.org/) and mzML (http://www.psidev.info/mzml) and is made available through the respective websites. The validator is also released as open source under Mozilla Public License 2.0 at https://gitlab.com/imzML/imzMLValidator.

Introduction

The open mass spectrometry imaging (MSI) format imzML has enjoyed a significant increase in uptake since its inception.[1] Its introduction has enabled previously impossible, or impractical, scientific endeavours to be undertaken such as multicentre studies,[2] data repositories for MSI data, such as PRIDE,[3] and data analysis platforms, such as METASPACE.[4] In addition to this, a wide variety of software tools, both open source and commercial, for MSI data visualisation and analysis have since been released based either purely upon, or supporting the use of, imzML.[5,6,7,8,9,10,11] This has improved access to a larger array of data processing, statistical and multivariate analysis methods as the user is no longer limited to the features available in the proprietary software accompanying the

instrumentation. It is now possible to use multiple imzML software tools to analyse the same data, allowing researchers to select the most appropriate tool for the task at hand.

The imzML format is based on the open mass spectrometry format mzML.[12] The major deviations from the mzML specification are the separation of the storage of metadata (imzML file; header data stored as an XML file) and experimental data (ibd file; stored as a binary file), and the incorporation of imaging specific metadata. The imzML header file uses the same style and structure as mzML to describe the experimental data itself as well as capturing metadata about the experiment, such as instrument configuration and sample preparation. Metadata in the mzML and imzML formats are included as so called controlled vocabulary (CV) parameters (as the name implies, these parameters are taken from one, or more, controlled vocabularies). Controlled vocabularies are a set of agreed upon terms, and corresponding definitions, usually focused around a specific topic or theme. As the names and definitions of the terms within the vocabulary have been agreed upon and published, they provide an unambiguous means of describing something that falls within the topic of the CV, in this case a mass spectrometry (imaging) experiment. Both the mzML format and the imzML format have dedicated ontologies for describing mass spectrometry terms and specifically mass spectrometry imaging related terms respectively. The ibd file contains the experimental data for efficient storage and handling of datasets with large numbers of spectra, which are characteristic of MSI experiments. This typically results in a significant reduction in size when compared to mzML, making imzML more suitable for processing larger datasets.

The number of tools available for visualising, processing and analysing imzML data is still increasing and most major instrument vendors (including Thermo Fisher Scientific, Bruker, Waters and Shimadzu) now support direct export to imzML from their proprietary formats. This community and industry support has led to imzML becoming the standard of choice for sharing, processing and storing MSI data for many research groups and institutions.

Unfortunately no software (and also no user of software) is perfect, meaning that a user may experience the inability to load and process a dataset stored in the imzML format. Often it is not obvious to the user whether this is an issue with the software that produced the imzML file, an issue with the software loading the data, or a corrupted imzML file itself. Due to the complexity of the imzML specification, a complete implementation is non-trivial. This has resulted in varying levels of adherence to the specification in different implementations (meaning that features present in one implementation may not be present in another), as well as the introduction of some conventions that have spread across multiple implementations but are neither part of the standard nor ubiquitous. To users this can manifest in the inability to load datasets, incorrectly or inaccurately displayed data or, in severe cases, software crashes.

The majority of these incompatibilities are related to the varying levels of adherence to the standard of any given converter, exporter or producer of imzML files. These are often mitigated against through the use of dedicated validation tools. One such tool has been developed for the mzML standard.[13] This provides a good basis for checking the validity of an imzML file (as the original imzML specification stated that all imzML files should be valid mzML files), however it does not check any imzML specific properties, such as the additional imzML mapping rules and controlled vocabulary terms, or the presence and integrity of the binary data..

However, in part, such incompatibilities can also be related to the implementation of any given parser, reader or consumer of imzML files and whether this makes assumptions that are not covered in the specification or includes all possible features and functionality described therein. One challenge that can affect maintaining a complete implementation is the fact that the controlled vocabularies are updated to reflect the latest advances within the field. Typically updates are in the form of additional terms, the majority of which do not affect reading or writing (i)mzML, such as terms related to new instrumentation or data analysis algorithms. However, in rare cases, new terms can be introduced which describe new ways to store binary data, such as new compression schemes.[14] It is the ongoing responsibility of the software developers to support (or notify the user of a lack thereof) such new features as the format and controlled vocabularies develop. This process can be assisted by the publishing of up to date example files. The existing imzML example files have been updated to be fully compliant with the latest version of the imzML standard.

There are few imzML datasets that incorporate all metadata necessary to completely describe an experiment, with the metadata included often covering just enough to reconstruct the data (as either mass spectra or mass spectrometry images) successfully. This is already a significant improvement over sharing data as simple images. However, the full potential for the use of imzML is, at present, largely unrealised. The design of the imzML standard was such that detailed descriptions of the sample, sample preparation, instrument setup and experimental parameters could also be included.[1] In the proteomics field, guidelines for minimum information about a proteomics experiment (MIAPE) have been published, with a semantic validator for mzML to go along with it to check whether these are included within the metadata.[12,15] A discussion about the minimum reporting for MS imaging data has been in progress for a number of years and remains an ongoing project, with a proposal published in 2015.[16] If this becomes a requirement for either the publication or data repository submission of MSI data, it will become increasingly necessary for there to be a validator which can ensure that the reporting guidelines are met prior to submission.[3] An added benefit of having such metadata alongside the experimental data is an improved ability to organise, locate and identify data within an institution. This then provides the basis for conducting metadata analysis and meta-studies as well as being increasingly important as the field moves towards ‘big data’ inspired approaches for data mining, such as deep learning, where metadata can play a significant role.[17,18]

A contributing factor to the limited inclusion of metadata described above is the fact that specialised tools for editing the mzML or imzML metadata do not currently exist. Manual editing of an (i)mzML document is possible, but is not an easy process, especially when inserting new metadata. To do this, the user must open the file within a text editor, have a sufficient understanding of XML to not corrupt the structure, locate the most appropriate place in the (i)mzML structure to include the metadata, look up within the 2-4 OBO ontologies for the specific ontology reference for the metadata they would like to include and then insert the data in the correct (i)mzML format. It is also possible to use the *imzMLConverter* tool to add in metadata to an imzML file using a more intuitive interface, but this requires a complete ‘conversion’ (which will write out the entire binary data into another file) and so is not practical, especially for large datasets.[19]

Here we present a new software tool that performs validation on both mzML and imzML files, with a built-in, user-friendly metadata editor to enable users to resolve issues identified through the validation process. We also present a means of validating against complex sets of

metadata requirements, with the specific example of checking the proposed minimum reporting guidelines for MSI.

Materials and Methods

The validation tool is written in Java and comes in two forms, a command line interface (CLI) tool and graphical user interface (GUI) tool. The CLI tool includes only validation, where the result is output either to plain text or as JSON (JavaScript Object Notation) for ease of incorporation into existing automated workflows. It could, for example, be integrated into the submission process of repositories such as PRIDE[3] or METASPACE[4].

The imzML validator presented here is endorsed by the imzML working group and HUPO-PSI for validation of imzML (http://imzml.org/) and mzML (http://www.psidev.info/mzml) and is made available through the respective websites. The validator is also released as open source under Mozilla Public License 2.0, with source code and an executable version available from https://gitlab.com/imzML/imzMLValidator.

Results and Discussion

The graphical user interface (GUI) version of the validator tool includes both validation (to check that the file is correct and can be visualised correctly) and editing (for fixing any issues preventing it from being visualised correctly). The interface is shown in Figure 1. On selection of an imzML file, the GUI tool will parse the imzML (displaying any errors encountered during parsing in the 'issue list') and upon completion will display the total ion current (TIC) image in the upper left corner. If the TIC image is displayed, this indicates that the file is sufficiently well formed that it should load successfully in most software tools. Inspection of the TIC image can also provide the user with a rapid and easy way to identify experimental artefacts (such as gradient or raster effects) as well as issues such as incomplete conversion (a portion of the TIC missing). Additional identifying information about the loaded imzML file can be found within the 'Report' section of the interface (middle left). By default this includes key metadata such as the pixel size, the number of pixels and the format the data is stored in, however this can be altered by the user in the configuration file of the validator dependent on their exact requirements.

The bottom left panel contains a visualisation of the entire imzML file, displayed as an expandable tree. This tree can be searched using the filter box directly above, enabling a fast and direct way to locate specific metadata without requiring knowledge of the imzML structure. Validation is performed by clicking on the 'Validate' button, and is described in detail below. Editing can be performed by either selecting a node from the tree view of the imzML file or by selecting an issue from the 'issue list'. If an issue is selected from this list, then the editor will be opened to edit the offending imzML tag. Editing is discussed in more detail in the 'Editing' section.

Validation

The primary function of the tool is to enables users to identify any issues with their data which may prevent correct visualisation. This is achieved by the validation of the imzML specification. The validation process is described in detail in the following section. Subsequent sections then describe additional options for validation, specifically validation

against the proposed minimum reporting guidelines for MSI[16] and validation specific to the experimental set-up or the sample analysed.

imzML specification validation

The process of validating an imzML file consists of three stages. First the imzML header file is validated against the imzML XML schema definition (which defines the structure and layout of the file), then the header file is validated against the mapping files (the process of which is described below) and finally the link between the imzML and the ibd file is tested (by validating that the UUID in the imzML file matches that in the ibd file). The integrity of the binary data file can optionally be checked by performing the hash function on the ibd file and comparing the result to the value in the imzML file (see the accompanying user manual for details). Checking the link between the two files as well as the integrity of the ibd file ensures that the binary data has not been altered since it was written.

Mapping files describe which controlled vocabulary (CV) parameters can, should and must be present within the imzML header file, and where they should be located within the imzML structure. The CV terms describing each parameter are taken from one or more ontologies stored in the OBO format. The three levels of CV inclusion (MUST, SHOULD and MAY) are described below.

(a) The given CV must be included within the imzML file to be fully conforming to the specification (MUST).
(b) The given CV should be included however there may exist reasons why these can be ignored but their omission and any subsequent implications of their omission have been considered thoroughly (SHOULD).
(c) The given CV may or may not be included at the discretion of the user (MAY).

The interface after checking an invalid file is shown in Figure 1, with the colour coding denoting the severity of the issue that has occurred. Red indicates that a severe error has occurred, such as the imzML file not conforming to the imzML XML schema or a MUST rule of the mapping file being violated. These are predominantly caused by the conversion or exporting software and may result in the incorrect display of, or the inability to load and process, data. In general, if the error message states “Invalid (i)mzML” or “Invalid format” then a violation of the specification has been detected, which indicates the converter/exporter has likely produced invalid imzML. In some cases, such as the presence of invalid (negative) binary offset locations, the validator tool automatically fixes the issues during the initial loading of the imzML file, so the user must simply ‘save’ the file again (using either the ‘Save’ or ‘Save As’ button visible in Figure 1) to produce valid imzML. An example of this issue can be found in the published imzML dataset MTBLS289 on the MetaboLights data repository. When opening this dataset in an imzML-compatible software tool, only part of the data is visible (only the spectra that have valid offsets) shown in Figure 2 (top left). Opening this dataset in the validator displays an error (of the form “Invalid offset -2414 at spectrum:id=13902”) for each spectrum that has an invalid offset associated with it, shown in Figure 2 (right). The user is notified that the validator can resolve this issue by the text “Attempted to fix by adding 2^32 to offset” below each error. Once the user has saved the fixed imzML, and opened the new file into the same imzML-compatible software tool, the entire data will be visible (Figure 2, bottom left). If the error states that there has been a “Mapping file error” then the controlled vocabulary (CV) parameters describing the metadata

are either missing or incorrect. These errors can be resolved by editing the metadata as discussed in the 'Editing' section.

Orange indicates that a SHOULD mapping rule has been violated. Yellow indicates that either a MAY rule has been violated or that certain CV parameters have been included in a part of the imzML document that was not expected. These are typically represented as "Combination failure" or "Unexpected CVParam(s)" errors and can be remedied by editing the metadata as above. The omission of SHOULD or MAY mapping parameters is not detrimental to the display or processing of the imzML data and should not cause errors when using any of the imzML processing tools, but may indicate that useful metadata is missing.

Conditional validation

By default only the mapping files which are part of the (i)mzML specification are used for the validation process and this therefore ensures that the file is valid with respect to the published (i)mzML technical specification. However, in some situations it may be additionally beneficial to validate against an extended set of constraints. These could be institutional requirements for metadata such as including sample reference numbers and the operator name and contact information, or could be minimum reporting requirements necessary for publication or submission to data repositories.

The inclusion of certain metadata may only be appropriate depending on the exact experimental set-up, the sample being analysed or a combination of both. For example, the choice of matrix and method of application is of importance when performing and reporting MALDI MSI, but is not necessary in DESI (which instead has other important properties necessary in reporting such as the solvent used). This kind of conditional validation is not supported by default by either of the mzML or imzML specifications and corresponding CV mapping files. The PSI semantic validator for mzML solved this issue by hard-coding multiple 'object rules' which are triggered, for example, on the presence of certain CV parameters.[13] While this provides fully flexible and customisable rules, the drawback of this approach is that the rules must be created or modified by someone proficient in Java each time the minimum reporting document is changed or when a new condition is required, such as the introduction of a new MS technique or workflow.

To achieve the desired level of flexibility for conditional validation, while also lowering the barrier of entry to creation and modification of these rules, an additional XML based conditional mapping file was developed as an alternative to the hard-coded approach. This file describes one or more conditions that, when met, will trigger validation against the specified mapping file, or set of included mapping rules. For example, the presence of the CV parameter 'MS:1000075' (matrix-assisted laser desorption ionization) will add a series of MALDI specific mapping rules to the rules to validate against, such as the requirement to include details of the matrix used, the concentration of the matrix and the solvent system. Simple conditions can be included within the XML file itself and if necessary more complex conditions can be included by writing a Java class which implements the Condition interface. Further details of the conditional mapping file, can be found in the user manual accompanying the validator. This provides both a user friendly method of including conditional mapping rules, as well as retaining the option for using the powerful flexibility of Java (in a similar way to the mzML validator[13]).

Reporting guidelines validation

A proposed set of guidelines for minimum reporting in MS imaging was published by McDonnell *et al.* as an output of the COST action.[16] These guidelines cover the entire MSI workflow from origin of the sample, through sample preparation and data acquisition, to display of the final results. As discussed previously, imzML is capable of storing metadata related to the experiment and this, combined with the fact that imzML is the data format of choice for sharing and publishing MSI data, makes imzML an ideal candidate for capturing the minimum reporting guidelines. We only propose the inclusion of sections which are relevant in the context of being included within imzML, namely up to and including the stage of data acquisition (and if appropriate, any data pre-processing applied prior to writing the data to imzML). Subsequent points in the proposed guidelines cover visualisation of data and compound identification, neither of which fall within the remit of imzML as a data storage format and are therefore not included.

Despite the general nature of the reporting guidelines, some points are unavoidably specific to a certain experimental set-up (for example, that the laser parameters must be included, which would not be relevant to DESI). To demonstrate how the conditional validation functionality, described above, could be used, and how the minimum reporting guidelines for MSI could be checked, a custom conditional mapping file and custom ontology file were generated based on the minimum reporting guidelines discussion for MSI and are included alongside the validator. If the discussion on minimum reporting requirements for MSI continues and become a requirement of inclusion for publication, this conditional validation format could be a way to formalise and document these requirements in way that can be validated and enforced.

Experimental or sample-dependant validation

As described above, the minimum reporting guidelines were intended to be as general as possible for the field of mass spectrometry imaging, and as such are not tailored to any specific experimental set-up or application area. For some applications, the guidelines may not completely cover all required metadata or include sections which are unnecessary or even inappropriate. To demonstrate how users could tailor these reporting guidelines to specific applications, two application specific examples (food and bacteria) tailored to information necessary to our group for internal storage, sharing and processing are included alongside the validator. These demonstrate how other researchers could develop application specific, project specific or even sample specific sets of rules that can be validated against prior to archival or submission to internal data storage systems.

Editing

All metadata stored within the imzML header file can be modified using the editing interface, shown in Figure 3. This includes the modification of any CV parameter responsible for the violation of a mapping rule (MUST, SHOULD or MAY) in order to to rectify the corresponding issue. The appearance of the editing interface is dependent on the CV parameter being modified and is generated automatically based on the mapping files, including the new conditional mapping files described above, that have been specified by the user to validate against.

For a given (i)mzML element, a dropdown selection box is included within the editing interface for each MUST, SHOULD and MAY rule that exists within the set of mapping files being validated against. This dropdown box only includes appropriate ontology terms for the

given (i)mzML element and the specific rule. Rules which have not been satisfied will be displayed with a '✕' whereas satisfied rules are denoted by a '✓', providing the user with continuous feedback as to the validity of any given (i)mzML element.

Following modification of the metadata in the mzML or imzML file, the previous (i)mzML file can either be updated ('Save'), or a completely new file can be written ('Save As'). In the case of mzML, updating the previous file and writing a new file will cause the entire experimental binary data to be re-written alongside the metadata, as these are included within the same file. However with an imzML dataset, it is possible to only update the imzML file with the new metadata, as the binary data has not been changed. It is also possible to create a completely new file from the modified imzML data, but in this case both the imzML file and the ibd (binary) file must be written, requiring more disk space and taking more time to complete.

imzML specification Update

The imzML specification, as well as the accompanying example files, has been updated to address issues raised by the community since its original publication and to enable the validation process to better detect nonsensical imzML files. Full details can be found on the imzML website (https://imzML.org), but are outlined briefly below.

The major change made is related to the terms defining pixel size. In the controlled vocabulary two sequential terms were defined as *pixel size (IMS:1000046)* and *image shape (IMS:1000047)*. In some cases, the latter was erroneously used as *pixel size y (IMS:1000047)*. This discrepancy led to different implementations using the same term in different ways. The two terms have been adjusted to avoid ambiguity, resulting in *pixel size (x) (IMS:1000046)* and *pixel size y (IMS:1000047)*. This adds new functionality to the imzML standard (i.e. allowing rectangular, but non-square, pixels to be represented) and should be compatible with existing imzML files. In the situation where the pixel sizes in the *x* and *y* dimensions are equal (which is true in the majority of cases), it is sufficient to only provide the CV parameter *pixel size (x) (IMS:1000046)*.

In the original specification there were no strict definitions of the formats to be used to represent the UUID and hash value in the imzML file. This has led to different implementations, with some imzML exporters including curly braces around the UUID (which is how Microsoft GUIDs are often represented, for example "{1523c65a-4326-401e-9a52-e3326b401871}"), other exporters following the UUID specification directly,[20] and others following neither of these with some combination of using curly braces, capital letters, and omitting hyphen separators. Similarly, there is a difference in how some exporters represent the hash value in the imzML file. To ensure consistency, additions to the specification have been made, stating that both the hash and UUID are case insensitive, and the UUID value should be stored using hyphen separators (in the form 8-4-4-4-12) with no curly braces.

Additional modifications to the specification, which improve the ability to detect nonsensical imzML files, include making the tags <scanSettingsList> and <scanSettings> mandatory. Details about the image display, such as the number of pixels in each dimension and pixel size are stored as parameters within the <scanSettings> tag. The <scan> tag, which is part of each spectrum, contains the spatial location of each pixel. Previously, both

tags were optional and therefore could (theoretically) be omitted, while remaining a technically valid imzML file.

Conclusions

The validator and editor tool presented here fills a current gap in the imzML arsenal, enabling users to check that their data is error free and can be loaded and visualised correctly. If the file is not valid, then the tool provides a user friendly way to rectify major issues and add, edit and delete accompanying metadata. In certain cases, this can be done automatically. As the field develops, it is becoming increasingly important to publish data alongside publications (often through submission to data repositories). To ensure that each dataset has the required documentation, a set of minimum reporting guidelines for MSI was recently proposed by a COST initiative on MSI.[16] We demonstrate here how it is possible to include metadata, which fulfils these guidelines, in imzML and how to use the validator tool presented here to check that all guidelines are met. To do this, the minimum reporting guidelines were translated into a newly developed conditional mapping file, which can be used by the validator tool to check metadata requirements. As imzML is based on the HUPO-PSI mzML format for mass spectrometry data, all features discussed above can also be applied to data stored in the mzML format. This also provides users of mzML a means to both validate and edit metadata included within the file in a user friendly manner. The new validator tool will have an additional benefit for initiatives such as data repositories, allowing them to create conditional mapping files which fit any custom metadata requirements and then validate against them prior to accepting the data.[3]

Acknowledgement

The authors gratefully acknowledge financial support from Deutsche Forschungsgemeinschaft (DFG RO 3421/6-1) and TechnologieAllianzOberfranken (TAO).

Supporting Information Available

Document describing the relevant (i)mzML ontology terms for each point in the minimum reporting guidelines for MSI.

Manual for imzML Validator.

References

(1) Schramm, T.; Hester, A.; Klinkert, I.; Both, J.-P.; Heeren, R. M.; Brunelle, A.; Laprévote, O.; Desbenoit, N.; Robbe, M.-F.; Stoeckli, M.; Spengler, B.; Römpp, A. *Journal of Proteomics* **2012**, *75*, 5106–5110.
(2) Römpp, A.; Both, J.-P.; Brunelle, A.; Heeren, R. M. A.; Laprévote, O.; Prideaux, B.; Seyer, A.; Spengler, B.; Stoeckli, M.; Smith, D. F. *Analytical and Bioanalytical Chemistry* **2015**, *407*, 2329–2335.
(3) Römpp, A.; Wang, R.; Albar, J. P.; Urbani, A.; Hermjakob, H.; Spengler, B.;

Vizcaíno, J. A. *Analytical and Bioanalytical Chemistry* **2015**, *407*, 2027–2033.
(4) Palmer, A.; Phapale, P.; Chernyavsky, I.; Lavigne, R.; Fay, D.; Tarasov, A.; Kovalev, V.; Fuchser, J.; Nikolenko, S.; Pineau, C.; Becker, M.; Alexandrov, T. *Nature Methods* **2016**, *14*, 57–60.
(5) Race, A. M.; Palmer, A. D.; Dexter, A. J.; Steven, R. T.; Styles, I. B.; Bunch, J. *Analytical Chemistry* **2016**,
(6) Robichaud, G.; Garrard, K. P.; Barry, J. A.; Muddiman, D. C. *Journal of the American Society for Mass Spectrometry* **2013**, *24*, 718–721.
(7) Källback, P.; Nilsson, A.; Shariatgorji, M.; Andrén, P. E. *Analytical Chemistry* **2016**, acs.analchem.5b04603.
(8) Rübel, O.; Greiner, A.; Cholia, S.; Louie, K.; Bethel, E. W.; Northen, T. R.; Bowen, B. P. *Analytical Chemistry* **2013**, *85*, 10354–10361.
(9) Parry, R. M.; Galhena, A. S.; Gamage, C. M.; Bennett, R. V.; Wang, M. D.; Fernández, F. M. *Journal of the American Society for Mass Spectrometry* **2013**, *24*, 646–649.
(10) Klinkert, I.; Chughtai, K.; Ellis, S. R.; Heeren, R. M. A. *International Journal of Mass Spectrometry* **2014**, *362*, 40–47.
(11) Bemis, K. D.; Harry, A.; Eberlin, L. S.; Ferreira, C.; Van De Ven, S. M.; Mallick, P.; Stolowitz, M.; Vitek, O. *Bioinformatics* **2015**, *31*, 2418–2420.
(12) Martens, L. et al. *Molecular & cellular proteomics : MCP* **2011**, *10*, R110.000133.
(13) Montecchi-Palazzi, L.; Kerrien, S.; Reisinger, F.; Aranda, B.; Jones, A. R.; Martens, L.; Hermjakob, H. *Proteomics* **2009**, *9*, 5112–5119.
(14) Teleman, J.; Dowsey, A. W.; Gonzalez-Galarza, F. F.; Perkins, S.; Pratt, B.; Rost, H.; Malmstrom, L.; Malmstrom, J.; Jones, A. R.; Deutsch, E. W.; Levander, F. *Molecular & cellular proteomics : MCP* **2014**, *13*, 1537–1542.
(15) MIAPE, *Nature Biotechnology* **2007**, *25*, 887–893.
(16) McDonnell, L. A.; Römpp, A.; Balluff, B.; Heeren, R. M. A.; Albar, J. P.; Andrén, P. E.; Corthals, G. L.; Walch, A.; Stoeckli, M. *Analytical and Bioanalytical Chemistry* **2015**, *407*, 2035–2045.
(17) LeCun, Y.; Bengio, Y.; Hinton, G. *Nature* **2015**, *521*, 436–444.
(18) Thomas, S. A.; Race, A. M.; Steven, R. T.; Gilmore, I. S.; Bunch, J. *IEEE Symposium Series on Computational Intelligence* **2016**,
(19) Race, A. M.; Styles, I. B.; Bunch, J. *Journal of Proteomics* **2012**, *75*, 5111–5112.
(20) Leach, P.; Mealling, M.; Salz, R. RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace. 2005; http://www.ietf.org/rfc/rfc4122.txt.

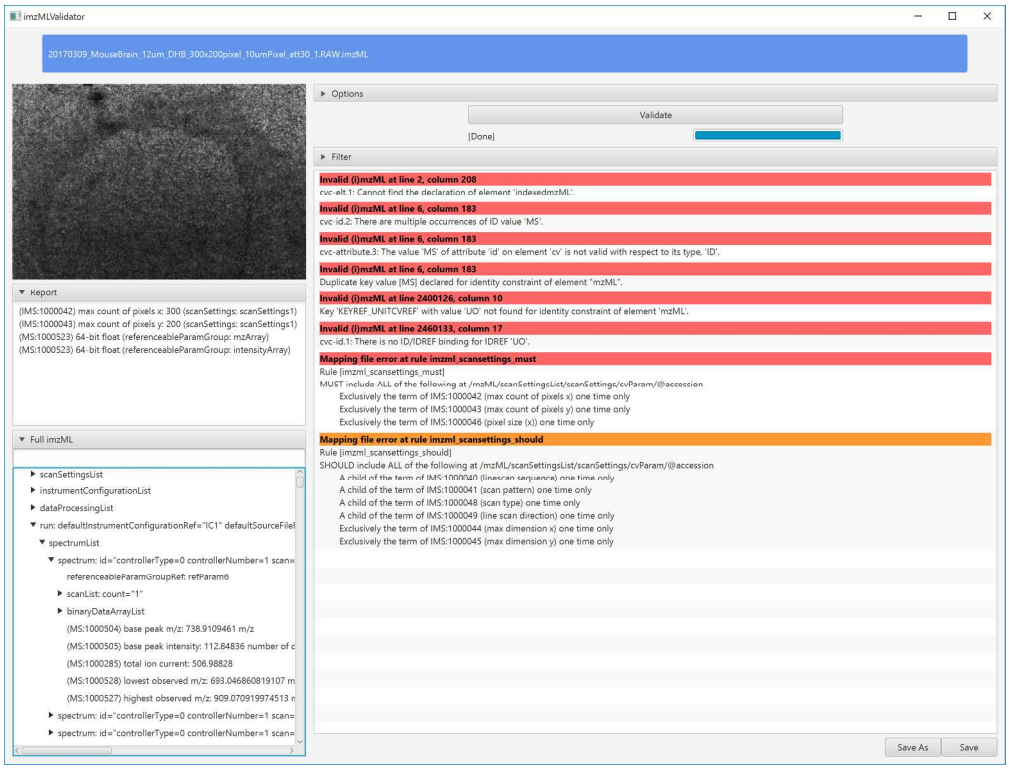


Figure 1: Interface for imzMLValidator after validation of an invalid imzML file. Total ion current (TIC) of the opened file is shown in the top left. An expandable tree representation of the imzML file (under the 'Full imzML' tab), with a search filter, is shown on the left side for ease of navigation. The main window shows the 'issue list' containing a series of colour coded descriptions of any violations of the specification encountered when performing validation. See text for a more detailed explanation.

353x268mm (192 x 192 DPI)

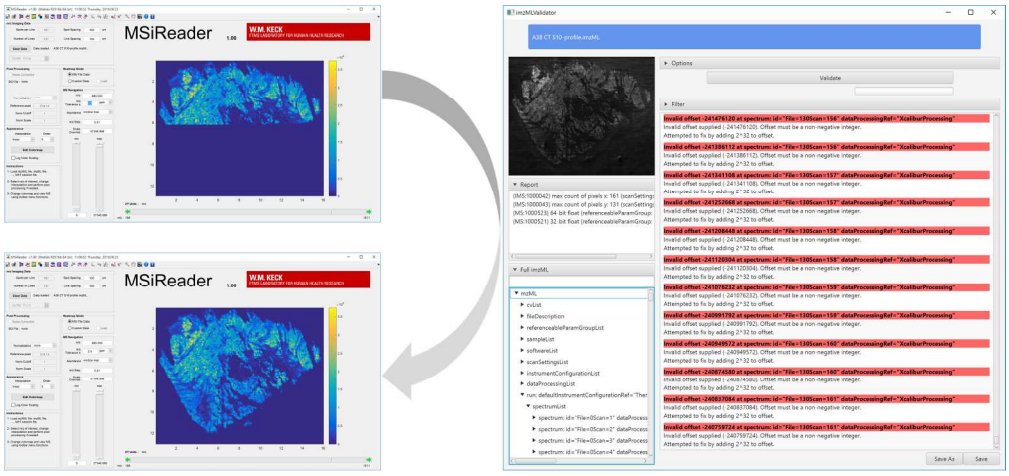


Figure 2: Some imzML files contain errors which prevent them from being successfully loaded and visualised in software tools supporting imzML. One such issue, is the use of invalid (negative) locations for the spectral data within the ibd file. In some software tools, this issue can be identified from the apparent missing data when interacting with the data (top left). Opening the problematic imzML file into the validator shows an error for each spectrum with the form "Invalid offset -2414 at spectrum:id=13902" (right). Saving the file using the validator and then reopening in the software of choice then results in the full data being visible (bottom left).

320x148mm (192 x 192 DPI)

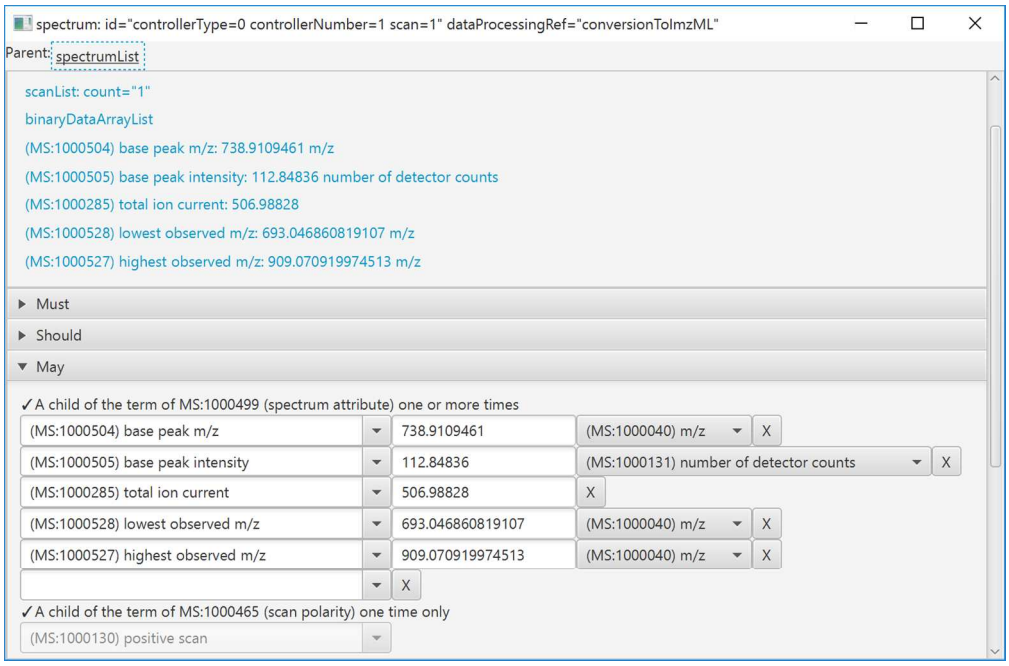
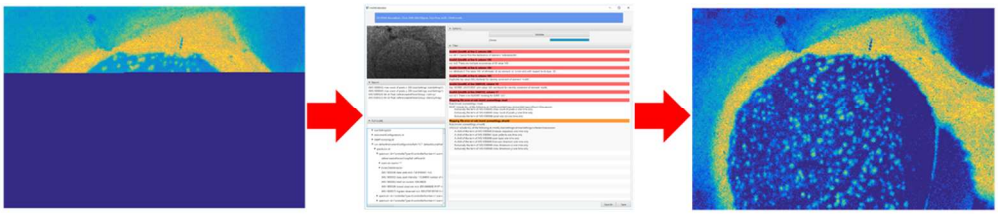


Figure 3: Interface for editing CV parameters on mzML tags. A list of all child parameters (including both imzML tags as well as controlled vocabulary parameters) for navigation is shown top. The MUST, SHOULD and MAY mapping rules are separated out into expandable dropdown boxes which include automatically generated entry boxes for each rule. Rules which have not been satisfied will be displayed with a '✖' whereas satisfied rules are denoted by a '✓'.

213x139mm (192 x 192 DPI)



Graphical abstract depicting the automatic resolution of an invalid imzML file and subsequent correct visualisation of data.

327x69mm (96 x 96 DPI)