PSI Draft

Johannes Griss, European Bioinformatics Institute
Timo Sachsenberg, University of Tübingen
Mathias Walzer, University of Tübingen
Oliver Kohlbacher, University of Tübingen
Henning Hermjakob, European Bioinformatics Institute
Juan Antonio Vizcaíno, European Bioinformatics Institute

March 2012

mzTab: exchange format for proteomics and metabolomics results

Status of This Document

This document presents a draft specification for the mzTab data format developed by members of the Human Proteome Organisation (HUPO) Proteomics Standards Initiative (PSI) Proteomics Informatics (PI) Working Group. Distribution is unlimited.

Version of This Document
The current version of this document is: version 0.9.0, release candidate March 2012.

Abstract

The Human Proteome Organisation (HUPO) Proteomics Standards Initiative (PSI) defines community standards for data representation in proteomics to facilitate data comparison, exchange and verification. The Proteomics Informatics Working Group is developing standards for describing the results of identification and quantification processes for proteins, peptides and protein modifications from mass spectrometry. This document defines a tab delimited text file format to report proteomics and metabolomics results.

Contents

Abstract........1
1. Introduction........2
1.1 Background........2
1.2 Document Structure........3
2. Use Cases for mzTab........3
3. Concepts and Terminology........4
4. Relationship to Other Specifications........4
4.1 The PSI Mass Spectrometry Controlled Vocabulary (CV)........5
5. Resolved Design and scope issues........6
5.1 Handling updates to the controlled vocabulary........6
5.2 Use of identifiers for input spectra to a search........6
5.3 Recommendations for reporting protein inference........7
5.4 Recommendations for reporting replicates within experimental designs........8
5.5 Recommendations for reporting quantification results........8
5.6 Encoding missing values, zeroes, nulls, infinity and calculation errors........9
5.7 Number of unique peptides reported........10
5.8 Reliability score........10
5.9 Reporting modifications........10
5.10 Comments on Specific Use Cases........11

5.11 Other supporting materials....12
6. Format specification....13
6.1 Sections....14
6.2 Metadata Section....14
6.3 Protein Section....19
6.4 Peptide Section....24
6.5 Small Molecule Section....27
7. Conclusions....31
8. Authors and Contributors....31
9. References....32
10. Intellectual Property Statement....32
Copyright Notice....32

1. Introduction

1.1 Background

This document addresses the systematic description of peptide, protein, and small molecule identification and quantification data retrieved from a mass spectrometry-based experiment. A large number of software tools are available that analyze MS data and produce a variety of different output data formats. The HUPO Proteomics Standards Initiative (PSI) has developed several vendor-neutral data formats to overcome this heterogeneity of data formats for MS data. Currently, the PSI promotes the usage of three file formats to report an experiment's data: mzML to store the pure MS data (i.e. the spectra and chromatograms), mzIdentML to store (poly)peptide identifications and potentially inferred protein identifications, and mzQuantML to store quantitative data associated with these results. All three of these formats are XML-based and require sophisticated software to access the stored data.

While full, detailed representation of MS data including provenance is essential for researchers in the field, many downstream analysis use cases are only concerned with the *results* of the experiment in an easily accessible format. In addition, there is a trend for performing more integrated experimental workflows involving both proteomics and metabolomics data. Thus, the current lack of standardization in the field of metabolomics was taken into account in the development of the here presented format, and structures were developed that can report protein, peptide, and small molecule MS based data.

mzTab is intended as a lightweight supplement to the already existing standard file formats, providing a summary, similar to the abstract of a scientific publication. mzTab files can contain protein, peptide, and small molecule identifications together with basic quantitative information. mzTab is not intended to store an experiment's complete data / evidence but only its final reported results. This format is also intended to provide local LIMS systems as well as MS proteomics repositories a simple way to share and combine basic information.

mzTab has been developed with a view to support the following general tasks (more specific use cases are provided in Section 2):

T1. *Facilitate the sharing of final experimental results,* especially with researchers outside the field of proteomics that i) lack specialized software to parse the existing PSI's XML-based standard file formats, and ii) are only interested in the final reported results and not in all the details related to the data processing due to the inherent complexity of MS proteomics data. Furthermore, this should encourage the development of small innovative tools without the requirement of parsing huge XML files, which might be outside the scope of many bioinformaticians.

T2. *Export of results to external software,* that is not able to parse proteomics/metabolomics specific data formats but can handle simple tab-delimited file formats. As guideline the file format is designed to be viewable by programs such as Microsoft Excel and Open Office Spreadsheet.

T3. *Contain the results of an experiment in a single file*, and thus not require linking two files to retrieve identification and quantification results to again simplify the processing of the data.

T4. *Allow the concatenation of results,* and thus being able to combine results from multiple experiments but also multiple entries from local LIMS databases or MS proteomics repositories.

T5. *Act as an output format of (web-) services* that report MS-based results and thus can produce standardized result pages.

T6. *Allow the combination of MS-based proteomics and metabolomics experimental results* within a single file.

T7. *Be able to link to the external experimental evidence* (i.e. the mass spectra in different formats), following the same approach used in mzIdentML and mzQuantML.

This document presents a specification, not a tutorial. As such, the presentation of technical details is deliberately direct. The role of the text is to describe the model and justify design decisions made. The document does not discuss how the models should be used in practice, consider tool support for data capture or storage, or provide comprehensive examples of the models in use. It is anticipated that tutorial material will be developed independently of this specification.

1.2 Document Structure

The remainder of this document is structured as follows. Section 2 lists use cases mzTab is designed to support. Section 3 describes the terminology used. Section 4 describes how the specification presented in Section 6 relates to other specifications, both those that it extends and those that it is intended to complement. Section 5 discusses the reasoning behind several design decisions taken. Section 6 contains the documentation of the file. Conclusions are presented in Section 7.

2. Use Cases for mzTab

The following use cases have driven the development of the mzTab data model, and are used to define the scope of the format in version 1.0.0.

1. mzTab files should be simple enough to make proteomics/metabolomics results accessible to people outside the respective fields. This should facilitate the sharing of data beyond the borders of the fields and make it accessible to non-experts. However, these files will not contain the complete evidence required to replicate the performed experiment or even provide information about all details of the reported identifications.
2. mzTab files should contain sufficient information to provide an electronic summary of all findings in a proteomics/metabolomics study to permit its use as a standard documentation format for ‘supplementary material’ sections of publications in proteomics and metabolomics. It should thus be able to replace PDF tables as a way of reporting peptides and proteins and make published identification and quantification information more accessible.

3. It should be possible to open mzTab files with "standard" software such as Microsoft Excel or Open Office Spreadsheet. This should furthermore improve the usability of the format to people outside the fields of proteomics/metabolomics.
4. It should be possible to export proteomics data from, for example, mzIdentML/ mzQuantML files into mzTab to then load this data into, for example, statistical tools such as those provided through the R programming language. With the current formats, complex conversion software would be needed to make proteomics results available to such environments.
5. mzTab files should make MS derived results easily accessible to scripting languages allowing bioinformaticians to develop software without the overhead of developing sophisticated parsing code. Since mzTab files will be comparatively small, the data from multiple experiments can be processed at once without requiring special resource management techniques.
6. It should be possible to contain the complete final results of an MS-based proteomics/metabolomics experiment in a single file. This should furthermore reduce the complexity of sharing and processing an experiment's final results. mzTab files should be able to store quantitative values for protein, peptide, and small molecule identifications. Furthermore, mzTab files should contain basic protein inference information and post-translational modification (PTM) position ambiguity information. Additionally, mzTab files should be able to report merged results from multiple search engines.
7. It should be possible to merge results from multiple experiments / resources by simply concatenating the respective sections of an mzTab file. Thus, every record in an mzTab file should be self-contained. However, it must be highlighted that quantitative results cannot be directly compared between different experiments.
8. It should be useful as an output format by web-services that can then be readily accessed by tools supporting mzTab. Through simple concatenation the results from multiple tools can be aggregated and processed at once.
9. As mzTab files only contain an experiment's core results, all entries should link back to their source. Furthermore, it should be possible to directly link a given peptide / small molecule identification to its source spectrum in an external MS data file. The same referencing system as in mzIdentML/mzQuantML should be used.

3. Concepts and Terminology

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" are to be interpreted as described in RFC-2119 (Bradner 1997).

4. Relationship to Other Specifications

The specification described in this document is not being developed in isolation; indeed, it is designed to be complementary to, and thus used in conjunction with, several existing and emerging models. Related specifications include the following:

1. *mzML* (http://www.psidev.info/mzml). mzML is the PSI standard for capturing mass spectra / peak lists resulting from mass spectrometry in proteomics (Martens, L., *et al.* 2011). mzTab files MAY be used in conjunction with mzML, although it will be possible to use mzTab with other formats of mass spectra. This document does not assume familiarity with mzML.
2. *mzIdentML* (http://www.psidev.info/mzidentml). mzIdentML is the PSI standard for capturing of peptide and protein identification data (Jones, A. R., *et al.* 2012). mzTab

files MAY reference mzIdentML files that then contain the detailed evidence of the reported identifications.

3. *mzQuantML* (http://www.psidev.info/mzquantml). mzQuantML is the proposed PSI standard for capturing quantitative proteomics data from mass spectrometry. mzTab files that report quantitative data MAY reference mzQuantML files for detailed evidence of the reported values.

4.1 The PSI Mass Spectrometry Controlled Vocabulary (CV)

The PSI-MS controlled vocabulary is intended to provide terms for annotation of mzML, mzIdentML, and mzQuantML files. The CV has been generated by collection of terms from software vendors and academic groups working in the area of mass spectrometry and proteome informatics. Some terms describe attributes that must be coupled with a numerical value attribute in the CvParam element (e.g. MS:1001191 “p-value”) and optionally a unit for that value (e.g. MS:1001117, “theoretical mass”, units = dalton). The terms that require a value are denoted by having a “datatype” key-value pair in the CV itself: MS:1001172 "mascot:expectation value" value-type:xsd:double. Terms that need to be qualified with units are denoted by have a “has_units” key in the CV itself (relationship: has_units: UO:0000221 ! dalton).

As recommended by the PSI CV guidelines, psi-ms.obo should be dynamically maintained via the psidev-ms-vocab@lists.sourceforge.net mailing list that allows any user to request new terms (via the form http://www.psidev.info/index.php?q=node/440) in agreement with the community involved. Once a consensus is reached among the community the new terms are added within a few business days. If there is no obvious consensus, the CV coordinators committee should vote and make a decision. A new psi-ms.obo should then be released by updating the file on the CVS server without changing the name of the file (this would alter the propagation of the file to the OBO website and to other ontology services that rely on file stable URI). For this reason an internal version number with two decimals (x.y.z) should be increased:

- x should be increased when a first level term is renamed, added, deleted or rearranged in the structure. Such rearrangement will be rare and is very likely to have repercussion on the mapping.
- y should be increased when any other term except the first level one is altered.
- z should be increased when there is no term addition or deletion but just editing on the definitions or other minor changes.

The following ontologies or controlled vocabularies specified below may also be suitable or required in certain instances:

- Unit Ontology (http://www.obofoundry.org/cgi-bin/detail.cgi?id=unit)
- ChEBI (http://www.ebi.ac.uk/chebi/)
- OBI (Ontology of Biological Investigations - http://obi.sourceforge.net/)
- PSI Protein modifications workgroup - http://psidev.sourceforge.net/mod/data/PSI-MOD.obo
- Unimod modifications database - http://www.unimod.org/obo/unimod.obo
- PRIDE ontology (http://ebi-pride.googlecode.com/svn/trunk/pride-core/schema/pride_cv.obo)

5. Resolved Design and scope issues

There were several issues regarding the design of the format that were not clear cut, and a design choice was made that was not completely agreeable to everyone. So that these issues do not keep coming up, we document the issues here and why the decision that is implemented was made.

5.1 Handling updates to the controlled vocabulary

There is a difficult issue with respect to how software should encode CV terms, such that changes to core can be accommodated. This issue is discussed at length in the mzML specification document (Martens, L et al. 2011), and mzTab follows the same convention. In brief, when a new term is required, the file producers must contact the CV working group (via the form http://www.psidev.info/index.php?q=node/440) and request the new term. It is anticipated that problems may arise if a consumer of the file encounters a new CV term and they are not working from the latest version of the CV file. It has been decided that rather than aim for a workaround to this issue, it can be expected that data file consumers must ensure that the OBO file is up-to-date.

5.2 Use of identifiers for input spectra to a search

Peptides and small molecules MAY be linked to the source spectrum (in an external file) from which the identifications are made by way of a reference in the spec_ref attribute and via the ms_file element which stores the URL of the file in the location attribute. It is advantageous if there is a consistent system for identifying spectra in different file formats. The following table is implemented in the PSI-MS CV for providing consistent identifiers for different spectrum file formats. This is the exact same approach followed in mzIdentML and mzQuantML. *Note, this table shows examples from the CV but will be extended. The CV holds the definite specification for legal encodings of spectrumID values.*

ID	Term	Data type	Comment

MS:1000768	Thermo nativeID format	controllerType=xsd:nonNegativeInteger
		controllerNumber=xsd:positiveInteger
		scan=xsd:positiveInteger.
			controller=0 is usually the mass spectrometer

MS:1000769	Waters nativeID format	function=xsd:positiveInteger
		process=xsd:nonNegativeInteger
		scan=xsd:nonNegativeInteger	

MS:1000770	WIFF nativeID format	sample=xsd:nonNegativeInteger
		period=xsd:nonNegativeInteger
		cycle=xsd:nonNegativeInteger
		experiment=xsd:nonNegativeInteger	

MS:1000771	Bruker/Agilent YEP nativeID format	scan=xsd:nonNegativeInteger	
MS:1000772	Bruker BAF nativeID format	scan=xsd:nonNegativeInteger	
MS:1000773	Bruker FID nativeID format	file=xsd:IDREF	The nativeID must be the same as the source file ID
MS:1000774	multiple peak list nativeID format	index=xsd:nonNegativeInteger	Used for conversion of peak list files with multiple spectra, i.e. MGF, PKL, merged DTA files. Index is the

			spectrum number in the file, starting from 0.
MS:1000775	single peak list nativeID format	file=xsd:IDREF	The nativeID must be the same as the source file ID. Used for conversion of peak list files with one spectrum per file, typically in a folder of PKL or DTAs, where each sourceFileRef is different
MS:1000776	scan number only nativeID format	scan=xsd:nonNegativeInteger	Used for conversion from mzXML, or a DTA folder where native scan numbers can be derived.
MS:1000777	spectrum identifier nativeID format	spectrum=xsd:nonNegativeInteger	Used for conversion from mzData. The spectrum id attribute is referenced.

Table 1 Controlled vocabulary terms and rules implemented in the PSI-MS CV for formulating the "nativeID" to identify spectra in different file formats.

In mzTab, the spec_ref attribute should be constructed following the data type specification in Table 1. As an example, to reference the third spectrum (index = 2) in an MGF (Mascot Generic Format) file:

```
MTD   UNIT_1-ms_file[1]-format     [MS, MS:1001062, Mascot MGF file, ]
MTD   UNIT_1-ms_file[1]-id_format  [MS, MS:1000774, multiple peak list nativeID format, ]
```

...

```
PEH   sequence   ...   spec_ref               ...
PEP   NILNELFQR  ...   ms_file[1]:index=2     ...
```

5.3 Recommendations for reporting protein inference

There are multiple approaches to how protein inference can be reported. mzTab is designed to only hold experimental results which in proteomics experiments can be very complex. At the same time, for down-stream statistical analysis there is a need to simplify this problem. It is not possible to model detailed protein inference data without a significant level of complexity at the file format level. Therefore, it was decided to "mention" the protein inference problem in mzTab files but not provide detailed information on how it was resolved. Protein entries in mzTab files contain the field ambiguity_members. The protein accessions listed in this field should identify proteins that could also be identified through the same (sub-)set of peptides but were not chosen as the primary identification. The members of the ambiguity group are not reported in the peptide table for the respective unit.

```
COM   In the following example only one peptide was identified that can be attributed to
COM   multiple proteins. The choice which one to pick as primary accession depends on the
COM   resource generating the mzTab file.
```

...

```
PRH   accession  unit_id   …   ambiguity_members                                 …
PRT   P19012     EXP_1         P13646, P08779, P02533, Q7Z3Z0, Q7Z3Y9, Q7Z3Y8    …
```

...

```
PEH   sequence       accession    unit_id …
PEP   ALEEANADLEVK   P19012       EXP_1   …
```

5.4 Recommendations for reporting replicates within experimental designs

Modeling the correct reporting of technical/biological replicates within experimental designs is inherently complex. mzQuantML supports the detailed reporting of such results in respect to quantitative data. mzTab is designed to be a simple data format. Therefore, the reporting of results from such experimental designs is not explicitly supported in mzTab.

However, mzTab allows the reporting of multiple experiments (called units) in one single file. Thus, it is possible for tools to export results from replicates of one experiment in a single file as different units. Depending on the use case it is also possible to only export the final results of a given experiment in the generated mzTab file, which could potentially be a unified view based on the different technical/biological replicates. The used method SHOULD be reported in the unit's description in the meta-data section.

The above ways to report results from replicates are only meant as examples but not recommendations. A recommendation of how to report replicates is deliberately not given here as it depends on the individual use case which approach is used.

5.5 Recommendations for reporting quantification results

There are multiple quantification techniques available for MS-based experiments that often result in slightly different types of data. mzTab was explicitly not designed to capture any of these specific differences. The goal for mzTab was to provide a generic view on quantitative MS-based identification data that is applicable to as many different quantitation methods as possible.

Quantitative technologies generally result in some kind of abundance measurement of the identified analyte. Several of the available techniques furthermore allow/require multiple similar samples to be multiplexed and analyzed in a single MS run. When several biological samples are multiplexed these samples are referred to as “subsamples” in mzTab. Subsamples MUST furthermore be linked to the used labels in the metadata section of the mzTab file (see example below). In case a quantification method is used that does not lead to multiplexed biological samples the generated quantification values MUST be reported as subsample 1. Replicates in experimental designs are not explicitly modeled in mzTab (see section 5.4).

mzTab allows the reporting of abundance, standard deviation, and standard error for any present subsample. The unit of these values MUST be specified in the metadata section of the mzTab file. This should be used to differentiate between relative (use unit “ratio”) and absolute (use respective unit) quantitation. The reported values SHOULD represent the final result of the performed data analysis. The exact meaning of the values will thus depend on the used analysis pipeline and quantitation method and is not expected to be comparable across multiple mzTab files.

In MS label-based quantitation approaches the final ratios of the various tags MUST be reported as the various subsample abundances at the peptide level. The final protein abundances MUST be reported at the protein level using the same subsample labels (see example below).

```
COM   The following example shows how two different quantitative experiments
COM   can be reported in one mzTab file. Not all labels are shown
…
MTD   EXP_1-quantification_method           [MS,MS:1001837,iTraq,]
MTD   EXP_1-sub[1]-description              Healthy human liver tissue
MTD   EXP_1-sub[1]-quantification_reagent   [PRIDE,PRIDE:0000114,iTRAQ reagent 114,]
MTD   EXP_1-sub[2]-description              Human hepatocellular carcinoma sample.
MTD   EXP_1-sub[2]-quantification_reagent   [PRIDE,PRIDE:0000115,iTRAQ reagent 115,]
…
MTD   EXP_2-quantification_method           [MS,MS:100999,SILAC,]
MTD   EXP_2-sub[1]-description              Healthy rat liver tissue
MTD   EXP_2-sub[1]-quantification_reagent   [PRIDE,PRIDE:0000325,SILAC heavy,]
MTD   EXP_2-sub[2]-description              Intoxicated rat liver.
MTD   EXP_2-sub[2]-quantification_reagent   [PRIDE,PRIDE:0000326,SILAC light,]
…
PRH   accession   unit_id   …   protein_abundance_sub[1]  …  protein_abundance_sub[2]  …
PRT   P12345      EXP_1     …   1                         …  0.82749
PRT   P15151      EXP_2     …   2.42114                   …  1
…
COM   All subsample columns where the SILAC based experiment has no values but the iTraq
COM   base experiment has, the SILAC experiment’s proteins simply contain “-“.
```

MS^2 spectral counting-based approaches SHOULD be reported using optional columns in the peptide table as well as the protein table as they only result in one single value per analyte. In case the approach used also generates standard deviation and standard errors the quantification results MAY also be reported using the subsample 1 columns. MS label-free quantitation techniques do not require any additional support in mzTab as they simply need to report abundance values per sample in a straight-forward manner. CV parameter accessions MAY be used as optional column names following the following format: opt_cv_{accession}.

```
COM   Example showing how emPAI values are reported in an additional column using
COM   MS CV parameter “emPAI value” (MS:1001905)
…
PRH   accession  …   opt_cv_MS:1001905
PRT   P12345     …   0.658
```

5.6 Encoding missing values, zeroes, nulls, infinity and calculation errors

In the table-based sections (protein, peptide, and small molecule) there MUST NOT be any empty cells. In case a given property is not available for an entry two symbols MUST be used to indicate why this information is not available:

- “-“ indicates that the given information was lost during conversion or not available in the source file (for example, modifications were not reported in the input file).
- “NA” should be given in case the information is not applicable (for example, a protein/ peptide does not contain any modifications but modifications are reported).

Any calculation that results in “not a number” (NaN type) MUST be reported using “NA” if the value could theoretically be calculated based on the available information. If a required value for the calculation was missing “-“ MUST be used. If ratios are included and the denominator is zero, the “INF” value MUST be given. In some cases, there is ambiguity with respect to these cases: e.g. in spectral counting if no peptide-spectrum matches are observed for a given protein, it is open for debate as to whether its abundance is zero or missing (“NA” as it could have been observed and “-“ if it could not have been measured).

5.7 Number of unique peptides reported

The protein section contains three columns to report the number of peptides supporting a given protein identification: num_peptides, num_peptides_distinct, num_peptides_unique (see below). The first column, num_peptides, reports all peptides that are associated with the given protein in the respective unit irrespective of their charge, modifications, and sequence. num_peptides_distinct MUST report the number of distinct peptides associated with the given protein-based on their sequence and modifications. num_peptides_unique finally reports the subset of distinct peptides (see above) that are unique for the given protein based on the protein database used for the performed search. Detailed examples on how these numbers are generated are given in the respective protein table column descriptions (see below). These numbers only reflect peptides used for identification but not quantification. The peptides contributing to the protein's quantification values can be retrieved from the peptide table by retrieving all peptides containing quantitative values assigned to the given proteins.

The idea of these three columns is to give the researcher a quick overview of how well a given protein identification is supported by peptide identifications. There will be cases where different tools generate different numbers for similar datasets. Nevertheless, too restrictive definitions of these columns might prevent the usage of mzTab for certain use cases.

5.8 Reliability score

All protein, peptide and small molecule identifications reported in an mzTab file SHOULD be assigned a reliability score (column "reliability" in all tables). The idea is to provide a way for researcher and/or MS proteomics or metabolomics repositories to score the reported identifications based on their own criteria. This score is completely resource-dependent and MUST NOT be interpreted as a comparable score between mzTab files generated from different resources. The criteria used to generate this score SHOULD be documented by the data providers. If this information is not provided by the producers of mzTab files, "-" must be provided as the value for each of the protein, peptide or small molecule identification.

The reliability MUST be an integer between 1-3 and SHOULD be interpreted as follows:

1: high reliability
2: medium reliability
3: poor reliability

The idea behind this score was to mimic the general concept of "resource based trust". For example, if PeptideAtlas reports identifications with a given reliability this would be interpreted differently as an identification reported from GPMDB. If resources now report their reliabilities using this metric and document how this metric is generated, a user can base his own interpretation of the results based on his trust in the resource. Furthermore, approaches to make various, for example search engine scores comparable have failed so far. To prevent the notion that the reported scores represent comparable probabilities this very abstract metric was chosen.

5.9 Reporting modifications

Modifications are modelled using a specific modification object with the following format: {position}{reliability score}-{Modification identifier}. {position} and {reliability score} are optional depending on the section where the modification is reported (for details see 6.3.15). Terminal modifications in proteins and peptides MUST be reported with the position set to 0 or the amino acid length +1 respectively. This object allows modifications to be assigned to ambiguous locations. Furthermore, it is possible to report reliabilities for every potential location.

For proteins and peptides modifications SHOULD be reported using either UNIMOD or PSI-MOD accessions. As these two ontologies are not applicable to small molecules, so-called CHEMMODs can also be defined. Two types of CHEMMODs are allowed: specifying a chemical formula or specifying a given *m/z* delta. The list of allowed modification identifiers is therefore:

```
CHEMMOD:+NH4
CHEMMOD:-18.0913
UNIMOD:18
PSI-MOD:00815
```

CHEMMODs MUST NOT be used for protein/peptide modifications if the respective entry is present in either the PSI-MOD or the UNIMOD ontology. Furthermore, mass deltas MUST NOT be reported if the given delta can be expressed through a known and unambiguous chemical formula.

To increase readability and make the information more accessible to people outside the field of proteomics, peptide modifications MAY be additionally defined in the peptide's sequence. Two approaches are allowed: specifying the modification name (preferably the PSI-MOD or UNIMOD name) in round brackets or specifying the *m/*z delta in square brackets after the affected amino acid. When using this approach, C-terminal modifications SHOULD be reported before the first amino acid. The ambiguity when reporting N-terminal modifications can only be resolved through the detailed information in the "modifications" column.

```
PEH  sequence                     … modifications …
PEP  EIEILAC(Carbamidomethyl)EIR  … 7-UNIMOD:4    …
PEP  EIEILAC[57.021464]EIR        … 7-UNIMOD:4    …
```

Irrespective of whether this approach is used or not the modifications MUST always be specified in the "modifications" column (see below). Therefore, a parser SHOULD ignore any values in a peptide's sequence within '[]' and within '()' and only retrieve modification data from the "modifications" column.

5.10 Comments on Specific Use Cases

Many special use cases for mzTab were considered during its development. Each of these use cases has a corresponding example file that exercises the relevant part of the format and provides a reference implementation example (see supporting documentation). Authors of software that create mzTab are encouraged to examine the examples that accompany this format release before implementing the writer.

5.10.1 Multiple database search engines

Proteomics groups now commonly analyze MS data using multiple search engines and combine results to improve the number of peptide and protein identifications that can be made. The output of such approaches can be represented in mzTab as follows: mzTab files SHOULD only contain the "final" protein list generated by any such workflow. Any protein, peptide, and small molecule can be associated with any number of search engines as well as multiple search engine scores. Thus, it is possible to report which element was identified by which search engine together with the resulting scores.

5.10.2 Merging mzTab files

All sections of an mzTab file were designed that they can easily be merged by simply concatenating the information. When merging mzTab files the rows of every section should be concatenated. For the table-based sections (proteins, peptides, and small molecules) the header row MUST only be reported once at the top of the section. Custom columns MUST NOT be included in merged files unless it was made sure that these columns report identical information.

5.10.3 Adding optional columns

Additional columns can be added to the end of rows in all the table-based sections (protein, peptide, and small molecule). These column headers MUST start with the prefix “opt_”. Column names MUST only contain the following characters: ‘A’-‘Z’, ‘a’-‘z’, ‘0’-‘9’, ‘_’, ‘-’, ‘[’, ‘]’, and ‘:’.

The information stored within an optional column is completely up to the resource that generates the file. It MUST not be assumed that optional columns having the same name in different mzTab files contain the same type of information. CV parameter accessions MAY be used as optional column names according to the following convention: opt_cv_{accession}.

```
COM   Example showing how emPAI values are reported in an additional column using
COM   MS CV parameter “emPAI value” (MS:1001905)
…
PRH   accession  …   opt_cv_MS:1001905
PRT   P12345     …   0.658
```

5.10.4 Referencing external resources (i.e. mzIdentML or mzQuantML files)

In mzTab all identifications SHOULD reference external resources that contain detailed evidence for the identification. This link MUST be stored in the “uri” column of the respective table. This field MUST NOT be used to reference an external MS data file. MS data files should be referenced using the method described in Section 5.2.

Where these URIs point to depends on the resource that generated the mzTab file. If, for example, PeptideAtlas was exporting data in the mzTab format the URI would be expected to point to the identification’s entry within the respective PeptideAtlas build. mzTab files originating from an mzIdentML files MAY reference the mzIdentML file using the URI column. In case quantitative values are reported coming from an mzQuantML file, the mzQuantML file SHOULD be referenced as it contains the reference to the underlying mzIdentML file.

5.11 Other supporting materials

The following example instance documents are available and between them cover all the use cases supported.

All example files can be downloaded from:
http://code.google.com/p/mztab/wiki/ExampleFiles

a) mztab_SILAC_example.txt - (hand crafted) mzTab file showing how SILAC data can be reported.

b) mztab_itraq_example.txt - (hand crafted) mzTab file showing how iTRAQ data can be reported.
c) mztab_merged_example.txt - merged version of the example file a and b.
d) PRIDE_Exp_Complete_Ac_16649.xml-mztab.txt - file generated using the mztab-exporter (converted PRIDE experiment accession 16649) containing iTRAQ data.
e) mztab_lipidomics_example.txt – Example containing MS lipidomics data produced by the Lipid Data Analyzer tool (http://genome.tugraz.at/lda/lda_download.shtml).

6. Format specification

This section describes the structure of an mzTab file.

- **Field separator**
  The column delimiter is the Unicode Horizontal Tab character (Unicode codepoint 0009).
- **File encoding**
  The UTF-8 encoding of the Unicode character set is the preferred encoding for mzTab files. However, parsers should be able to recognize commonly used encodings.
- **Case sensitivity**
  All column labels and field names are case-sensitive.
- **Line prefix**
  Every line in an mzTab file must start with a three letter code identifying the type of line delimited by a Tab character. The three letter codes are as follows:
  - MTD for metadata
  - PRH for the protein table header line (the column labels)
  - PRT for rows of the protein table
  - PEH for the peptide table header line (the column labels)
  - PEP for rows of the peptide table
  - SMH for small molecule table header line
  - SML for rows of the small molecule table
  - COM for comment lines
- **Dates**
  Dates and times MUST be supplied in the ISO 8601 format (“YYYY-MM-DD”, “YYYY-MM-DDTHH:MMZ” respectively).
- **Decimal separator**
  In mzTab files the dot (“.”) MUST be used as decimal separator. Thousand separators MUST NOT be used in mzTab files.
- **Comment lines and empty lines**
  Comment lines can be placed anywhere in an mzTab file. These lines must start with the three-letter code COM and are ignored by most parsers. Empty lines can also occur anywhere in an mzTab file and are ignored.
- **Params**
  mzTab makes use of CV parameters. As mzTab is expected to be used in several experimental environments where parameters might not yet be available for the generated scores etc. all parameters can either report CV parameters or user parameters that only contain a name and a value.
  Parameters are always reported as [CV label, accession, name, value]. Any field that is not available should be left empty.

```
[MS, MS:1001207, Mascot,]
```

```
[MS, MS:1001171, Mascot:score, 40.21]
[,,A user parameter, The value]
```

- **Unit IDs**
  To link identifications to certain bits of metadata identifications are group into so-called "units". In a unit, a specific protein must be unambiguously identified by its accession number. For PRIDE generated data a unit would, for example, be an experiment while for PeptideAtlas generated data a unit would be a specific build.
  UNIT_IDs SHOULD consist of the resource identifier plus the resources internal unit id. A resource is anything that is generating mzTab files. UNIT_IDs do not have to be unique across multiple mzTab files. Duplication of UNIT_IDs MUST be prevented when merging mzTab files.
  UNIT_IDs MUST only contain the following characters: ‘A’-‘Z’, ‘a’-‘z’, ‘0’-‘9’, and ‘_’.
- **Subsample IDs**
  In several experimental approaches biological samples are multiplexed and analyzed in one single experiment. To be able to supply metadata specific to such a subsample, subsample ids in the format sub[1-n] are used.

```
MTD    PRIDE_1234-sub[1]-species    [NEWT, 9606, Homo sapiens (Human), ]
```

6.1 Sections

mzTab files can contain four different sections. All of these sections are optional. The metadata section is made up of key-value pairs. The other three sections, protein, peptide, and small molecule section are table-based.

6.2 Metadata Section

The metadata section can provide additional information about the dataset(s) reported in the mzTab file. All fields in the metadata section are optional. The fields in the metadata section should be reported first in order of the UNIT_IDs then in the order the various fields listed here. The field’s name and value MUST be separated by a tab character:

```
MTD    PRIDE_1234-species      [NEWT, 9606, Homo sapiens (Human),]
MTD    PRIDE_1234-publication  [PRIDE, PRIDE:00000029, PubMed, 12345]
```

In the following list of fields any term encapsulated by {} is meant as a variable which MUST be replaced accordingly.
The multiplicity numbers given in the descriptions below refer to one unit. For example, title MAY only be specified once per unit.

6.2.1 {UNIT_ID}-title

Description:	The unit’s human readable title.
Type:	String
Multiplicity:	0 .. 1
Example:	`MTD   PRIDE_1234-title   My first test experiment`

6.2.2 {UNIT_ID}-description

Description:	The unit’s human readable description.
Type:	String
Multiplicity:	0 .. 1

Example:	MTD PRIDE_1234-description An experiment investigating the effects of Il-6.

6.2.3 {UNIT_ID}-sample_processing[1-n]

Description:	A list of parameters describing a sample processing step. The order of the data_processing items should reflect the order these processing steps were performed in. If multiple parameters are given for a step these should be separated by a “|”.
Type:	Parameter List
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-sample_processing[1] [SEP, SEP:00173, SDS PAGE,] MTD PRIDE_1234-sample_processing[2] [SEP, SEP:00142, enzyme digestion,]|[MS, … MS:1001251, Trypsin,]

6.2.4 {UNIT_ID}-instrument[1-n]-source

Description:	The instrument's source used in the experiment. Multiple instruments are numbered 1..n
Type:	Parameter
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-instrument[1]-source [MS, MS:1000073, ESI,] … MTD PRIDE_1234-instrument[2]-source [MS, MS:1000598, ETD,]

6.2.5 {UNIT_ID}-instrument[1-n]-analyzer

Description:	The instrument’s analyzer type used in the experiment. Multiple instruments are enumerated 1..n
Type:	Parameter
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-instrument[1]-analyzer [MS, MS:1000291, linear ion trap,] … MTD PRIDE_1234-instrument[2]-analyzer [MS, MS:1000484, orbitrap,]

6.2.6 {UNIT_ID}-instrument[1-n]-detector

Description:	The instrument's detector type used in the experiment. Multiple instruments are numbered 1..n
Type:	Parameter
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-instrument[1]-detector [MS, MS:1000253, electron multiplier,] … MTD PRIDE_1234-instrument[2]-analyzer [MS, MS:1000348, focal plane collector,]

6.2.7 {UNIT_ID}-software[1-n]

Description:	Software used to analyze the data and obtain the results reported. The parameter’s value SHOULD contain the software’s version. The order (numbering) should reflect the order in which the tools were used.
Type:	Parameter
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-software[1] [MS, MS:1001207, Mascot, 2.3]

	MTD PRIDE_1234-software[2] [MS, MS:1001561, Scaffold, 1.0]

6.2.8 {UNIT_ID}-false_discovery_rate

Description:	The unit's false discovery rate(s) reported at the peptide and/or protein level. Multiple parameters MUST be separated by "|".
Type:	Parameter List
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-false_discovery_rate [MS, MS:1001364, pep:global FDR, 0.01]|… [MS, MS:1001214, prot:global FDR, 0.08]

6.2.9 {UNIT_ID}-publication

Description:	A publication on this unit. PubMed ids must be prefixed by "pubmed:", DOIs by "doi:". Multiple identifiers MUST be separated by "|".
Type:	String List
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-publication pubmed:21063943|doi:10.1007/978-1-60761-987-1_6 MTD PRIDE_1234-publication pubmed:20615486|doi:10.1016/j.jprot.2010.06.008

6.2.10 {UNIT_ID}-contact[1-n]-name

Description:	The contact's name. Several contacts can be given by indicating the number in the square brackets after "contact". A contact has to be supplied in the format [first name] [initials] [last name] (see example).
Type:	String
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-contact[1]-name James D. Watson … MTD PRIDE_1234-contact[2]-name Francis Crick

6.2.11 {UNIT_ID}-contact[1-n]-affiliation

Description:	The contact's affiliation.
Type:	String
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-contact[1]-affiliation Cambridge University, UK MTD PRIDE_1234-contact[2]-affiliation Cambridge University, UK

6.2.12 {UNIT_ID}-contact[1-n]-email

Description:	The contact's e-mail address.
Type:	String
Multiplicity:	0 .. *
Example:	MTD PRIDE_1234-contact[1]-email watson@cam.ac.uk … MTD PRIDE_1234-contact[2]-email crick@cam.ac.uk

6.2.13 {UNIT_ID}-uri

Description:	A URI pointing to the unit's source data (e.g., a PRIDE experiment or a PeptideAtlas built)
Type:	URI
Multiplicity:	0 .. *

Example:	`MTD PRIDE_1234-uri http://www.ebi.ac.uk/pride/url/to/experiment`

6.2.14 {UNIT_ID}-mod

Description:	A list of "|" separated parameters describing all (distinct) PTMs reported in this unit.
Type:	Parameter List
Multiplicity:	0 .. 1

Example:	`MTD PRIDE_1234-mod [MOD, MOD:00397, iodoacetamide derivatized residue, ]|…`
	`[MOD, MOD:00675, oxidized residue, ]`

6.2.15 {UNIT_ID}-mod-probability_method

Description:	Method used to report modification (position) probabilities.
Type:	Parameter
Multiplicity:	0 .. 1

Example:	`COM As no CV parameters exist to describe such methods, a user parameter was used`
	`COM for this example`
	`MTD PRIDE_1234-mod-probability_method [,,Ascore,]`

6.2.16 {UNIT_ID}-quantification_method

Description:	The quantification method used in this unit (most of the times experiment) (if any).
Type:	Parameter
Multiplicity:	0 .. 1
Example:	`MTD PRIDE_1234-quantification_method [MS, MS:1001837, iTraq, ]`

6.2.17 {UNIT_ID}-protein-quantification_unit

Description:	Defines what type of units is reported in the protein quantification fields.
Type:	Parameter
Multiplicity:	0 .. 1
Example:	`MTD PRIDE_1234-protein-quantification_unit [PRIDE, PRIDE:0000395, Ratio, ]`

6.2.18 {UNIT_ID}-peptide-quantification_unit

Description:	Defines what type of units is reported in the peptide quantification fields.
Type:	Parameter
Multiplicity:	0 .. 1
Example:	`MTD PRIDE_1234-peptide-quantification_unit [PRIDE, PRIDE:0000395, Ratio, ]`

6.2.19 {UNIT_ID}-ms_file[1-n]-format

Description:	A parameter specifying the data format of the external MS data file.
Type:	Parameter
Multiplicity:	0 .. *

Example:	`MTD PRIDE_1234-ms_file[1]-format [MS, MS:1000584, mzML file, ]`
	`…`
	`MTD PRIDE_1234-ms_file[2]-format [MS, MS:1001062, Mascot MGF file, ]`

6.2.20 {UNIT_ID}-ms_file[1-n]-location

Description:	Location of the external data file.
Type:	URL
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-ms_file[1]-location  file://C:\path\to\my\file`
	…
	`MTD  PRIDE_1234-ms_file[2]-location  ftp://ftp.ebi.ac.uk/path/to/file`

6.2.21 {UNIT_ID}-ms_file[1-n]-id_format

Description:	Parameter specifying the id format used in the external data file.
Type:	Parameter
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-ms_file[1]-id_format  [MS, MS:1001530, mzML unique identifier, ]`
	…
	`MTD  PRIDE_1234-ms_file[2]-id_format  [MS, MS:1000774, multiple peak list … nativeID format, ]`

6.2.22 {UNIT_ID}-custom

Description:	Any additional parameters describing the unit.
Type:	Parameter
Multiplicity:	0 .. *
Example:	`MTD  PRIDE_1234-custom  [,,MS operator, Florian]`

6.2.23 {UNIT_ID}(-{SUB_ID})-species[1-n]

Description:	The respective species. Multiple species can be supplied. If there were multiple subsamples analyzed in the respective unit these species should be given using the additional "–{SUB_ID}" part. Subsample specific parameters describing one sample should all contain the same number between the brackets.
Type:	Parameter
Multiplicity:	0 .. *

Example:	`COM  Experiment where all subsamples (if any) consisted of the same two species`
	`MTD  PRIDE_1234-species[1]  [NEWT, 9606, Homo sapiens (Human), ]`
	`MTD  PRIDE_1234-species[2]  [NEWT, 12059, Rhinovirus, ]`
	`COM  Experiment where different samples from different species (combinations)`
	`COM  where pooled in one single MS analysis.`
	`MTD  PRIDE_1235-sub[1]-species[1]  [NEWT, 9606, Homo sapiens (Human), ]`
	`MTD  PRIDE_1235-sub[1]-species[2]  [NEWT, 573824, Human rhinovirus 1, ]`
	`MTD  PRIDE_1235-sub[2]-species[1]  [NEWT, 9606, Homo sapiens (Human), ]`
	`MTD  PRIDE_1235-sub[2]-species[2]  [NEWT, 12130, Human rhinovirus 2, ]`

6.2.24 {UNIT_ID}(-{SUB_ID})-tissue[1-n]

Description:	The respective tissue. For detailed documentation see 6.2.23
Type:	Parameter
Multiplicity:	0 .. *
Example:	`MTD  PRIDE_1234-tissue[1]  [BTO, BTO:0000759, liver, ]`

6.2.25 {UNIT_ID}(-{SUB_ID})-cell_type[1-n]

Description:	The respective cell type. For detailed documentation see 6.2.23
Type:	Parameter
Multiplicity:	0 .. *
Example:	`MTD  PRIDE_1234-cell_type[1]  [CL, CL:0000182, hepatocyte, ]`

6.2.26 {UNIT_ID}(-{SUB_ID})-disease[1-n]

Description:	The respective disease. For detailed documentation see 6.2.23
Type:	Parameter
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-disease[1]  [DOID, DOID:684, hepatocellular carcinoma, ]`
	`MTD  PRIDE_1234-disease[2]  [DOID, DOID:9451, alcoholic fatty liver, ]`

6.2.27 {UNIT_ID}-{SUB_ID}-description

Description:	A human readable description of the subsample.
Type:	String
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-sub[1]-description  Hepatocellular carcinoma samples.`
	`MTD  PRIDE_1234-sub[2]-description  Healthy control samples.`

6.2.28 {UNIT_ID}-{SUB_ID}-quantification_reagent

Description:	The reagent used to label the given subsample.
Type:	Parameter
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-sub[1]-reagent  [PRIDE,PRIDE:0000114,iTRAQ reagent,114]`
	`MTD  PRIDE_1234-sub[2]-reagent  [PRIDE,PRIDE:0000115,iTRAQ reagent,115]`

6.2.29 {UNIT_ID}-{SUB_ID}-custom

Description:	Parameters describing the subsample’s additional properties.
Type:	Parameter
Multiplicity:	0 .. *

Example:	`MTD  PRIDE_1234-sub[1]-custom   [,,Extraction date, 2011-12-21]`
	`MTD  PRIDE_1234-sub[1]-custom   [,,Extraction reason, liver biopsy]`

6.3 Protein Section

The protein section is table-based. The protein section must always come after the metadata section if the metadata section is present in the file. All table columns MUST be tab-separated. There MUST NOT be any empty cells.
The columns in the protein section MUST be in the order they are presented in this document.

6.3.1 accession

Description:	The accession of the protein in the source database. A protein accession MUST be unique within one UNIT. Together with the UNIT_ID the accession creates a unique identifier within a given mzTab file.
Type:	String

Example:	`PRH  accession  …`
	`PRT  P12345     …`

	PRT P12346 …

6.3.2 unit_id

Description:	The unit the protein comes from. See section 0 for detailed information on how the unit_id should be generated.
Type:	String
Example:	PRH accession unit_id … PRT P12345 PRIDE_1234 … PRT P12346 PRIDE_1234 …

6.3.3 description

Description:	The protein's name and or description line.
Type:	String
Example:	PRH accession unit_id description … PRT P12345 PRIDE_1234 Aspartate aminotransferase, mitochondrial … PRT P12346 PRIDE_1234 Serotransferrin …

6.3.4 taxid

Description:	The NCBI/NEWT taxonomy id for the species the protein was identified in.
Type:	Integer
Example:	PRH accession … taxid … PRT P12345 … 10116 … PRT P12346 … 10116 …

6.3.5 species

Description:	The human readable species the protein was identified in (this should be the NCBI entry's name).
Type:	String
Example:	PRH accession … taxid species … PRT P12345 … 10116 Rattus norvegicus (Rat) … PRT P12346 … 10116 Rattus norvegicus (Rat) …

6.3.6 database

Description:	The protein database used for the search (could theoretically come from a different species). Wherever possible the Miriam (http://www.ebi.ac.uk/miriam) assigned name SHOULD be used.
Type:	String
Example:	PRH accession … taxid species database … PRT P12345 … 10116 Rattus norvegicus (Rat) UniProtKB … PRT P12346 … 10116 Rattus norvegicus (Rat) UniProtKB …

6.3.7 database_version

Description:	The protein database's version – in case there is no version available (custom build) the creation / download (e.g., for NCBI nr) date SHOULD be given.
Type:	String
Example:	PRH accession … taxid species database database_version … PRT P12345 … 10116 Rattus norvegicus (Rat) UniProtKB 2011_11 … PRT P12346 … 10116 Rattus norvegicus (Rat) UniProtKB 2011_11 …

6.3.8 search_engine

Description:	A “|” delimited list of search engine(s) used to identify this protein. Search engines MUST be supplied as parameters.
Type:	Parameter List
Example:	COM In this example the first protein was identified by Mascot and Sequest while COM the second protein was only identified by Mascot. PRH accession … search_engine … PRT P12345 … [MS,MS:1001207,Mascot,]|[MS,MS:1001208,Sequest,] … PRT P12346 … [MS,MS:1001207,Mascot,] …

6.3.9 search_engine_score

Description:	A “|” delimited list of search engine score(s) for the given protein. Scores SHOULD be reported using CV parameters whenever possible.
Type:	Parameter List
Example:	PRH accession … search_engine_score … PRT P12345 … [MS,MS:1001171,Mascot score,50]|[MS,MS:1001155,Sequest:xcorr,2] … PRT P12346 … [MS,MS:1001171,Mascot score,47.2] …

6.3.10 reliability

Description:	The reliability of the given protein identification. This must be supplied by the resource and has to be one of the following values: 1: high reliability 2: medium reliability 3: poor reliability Important: An identification's reliability is resource-dependent.
Type:	Integer
Example:	PRH accession … reliability … PRT P12345 … 3 … PRT P12346 … 1 …

6.3.11 num_peptides

Description:	The total number of peptides identifying this protein.
Type:	Integer
Example:	COM P12345 is identified through ABCM, ABCM+Oxidation, CDE, CDE … PRH accession … num_peptides … PRT P12345 … 4 …

6.3.12 num_peptides_distinct

Description:	The number of distinct peptides identifying this protein. Distinct peptides are defined based on their sequence + modifications.
Type:	Integer
Example:	COM P12345 is identified through ABCM, ABCM+Oxidation, CDE, CDE … PRH accession … num_peptides_distinct … PRT P12345 … 3 …

6.3.13 num_peptides_unambiguous

Description:	The number of unambiguous distinct (only fitting this protein in the used search database) peptides identifying this protein.

Type:	Integer
Example:	COM P12345 is identified through ABCM, ABCM+Oxidation, CDE, CDE COM ABCM is only from P12345, CDE from P12345 and P12346 … PRH accession … num_peptides_unambiguous … PRT P12345 … 2 …

6.3.14 ambiguity_members

Description:	A comma-delimited list of protein accessions. This field should be set in the protein considered the most likely of the ambiguity group (the protein identified through the accession in the first column). The accessions listed in this field should identify proteins that could also be identified through these peptides but were not chosen by the researcher or resource. The members of the ambiguity group are not reported in the protein table for the respective unit. The exact semantics of how the ambiguity members were defined depends on the resource.
Type:	String List
Example:	COM P12345, P12347, and P12348 can all be identified through the same peptides … PRH accession … ambiguity_members … PRT P12345 … P12347,P12348 …

6.3.15 modifications

Description:	A comma delimited list of modifications found in the given protein. Modifications have to be reported in the following format: {position in protein}{reliability score}-{Modification identifier} Modification location probabilities / abundances can optionally be supplied and MUST be represented as a Double between 0-1. In case the position of the modification is uncertain multiple positions can be supplied delimited by a "|". Furthermore, in case a position is unknown no position information MAY be supplied. Terminal modifications MUST be reported at position 0 or protein size + 1 respectively. Valid modification identifiers are either PSI-MOD or UNIMOD accession (including the “MOD:” / “UNIMOD:” prefix) or CHEMMODS. CHEMMODS have the format CHEMMOD:+/-{chemical formula or *m/z* delta}. Valid CHEMMODS are for example “CHEMMOD:+NH4” or “CHEMMOD:-10.1098”. CHEMMODs MUST NOT be used if the modification can be reported using a PSI-MOD or UNIMOD accession. Mass deltas MUST NOT be used for CHEMMODs if the delta can be expressed through a known chemical formula.
Type:	String
Example:	COM P12345 with MOD:00412@10 probability = 0.8 COM P12345 with MOD:00412@140 OR 145 with associated probabilities … PRH accession … modifications … PRT P12345 … 10[0.8]-MOD:00412,140[0.2]|145[0.3]-MOD:00412 …

6.3.16 uri

Description:	A URI pointing to the protein's source entry in the unit it was identified in (e.g., the PRIDE database or a local database / file identifier).
Type:	URI

Example:	PRT accession … uri … PRH P12345 … http://www.ebi.ac.uk/pride/url/to/P12345 …

6.3.17 go_terms

Description:	A comma-delimited list of GO accessions for this protein.
Type:	String List
Example:	PRT accession … go_terms … PRH P12345 … GO:0006457, GO:0005759, GO:0005886, GO:0004069 …

6.3.18 protein_coverage

Description:	A value between 0 and 1 defining the protein coverage.
Type:	Double
Example:	PRT accession … protein_coverage … PRH P12345 … 0.4 …

6.3.19 protein_abundance_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The protein's abundance in the given subsample. This information can only be interpreted when identifying the subsample's properties through the experiment id + subsample number in the metadata section of the file. The protein abundance reflects the protein's quantitative information after it was interpreted by the user as, for example when using peptide-based quantification methods like iTRAQ.
Type:	Double
Example:	PRT accession … protein_abundance_sub[1] … protein_abundance_sub[2] … PRH P12345 … 0.4 … 0.2 …

6.3.20 protein_abundance_stdev_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard deviation of the protein’s abundance. If a protein’s abundance is given for a certain subsample, the corresponding standard deviation column SHOULD also be present.
Type:	Double
Example:	PRT accession … protein_abundance_sub[1] protein_abundance_stdev_sub[1] … PRH P12345 … 0.4 0.05 …

6.3.21 protein_abundance_std_error_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard error of the protein’s abundance. If a protein’s abundance is given for a certain subsample, the corresponding standard error column MUST also be present.
Type:	Double
Example:	PRT accession … protein_abundance_sub[1] … protein_abundance_std_error_sub[1] … PRH P12345 … 0.4 … 0.03 …

6.3.22 opt_*

Description:	**Optional** (this column MAY be present)

	Additional columns can be added to the end of the protein table. These column headers MUST start with the prefix “opt_”. Column names MUST only contain the following characters: ‘A’-‘Z’, ‘a’-‘z’, ‘0’-‘9’, ‘_’, ‘-’, ‘[’, ‘]’, and ‘:’. CV parameter accessions MAY be used for optional columns following the format: opt_cv_{accession}.
Type:	Column

Example:	`PRT accession … opt_my_value opt_another_value`
	`PRH P12345 … My value some other value`

6.4 Peptide Section

The peptide section is table based. The peptide section must always come after the metadata section and or protein section if these are present in the file. All table columns MUST be Tab separated. There MUST NOT be any empty cells. The columns in the peptide section MUST be in the order they are presented in this document.

6.4.1 sequence

Description:	The peptide's sequence – can optionally contain modifications / delta masses in UNIMOD syntax (see section 5.9).
Type:	String

Example:	`PEH sequence …`
	`PEP KVPQVSTPTLVEVSR …`
	`PEP EIEILAC(Carbamidomethyl)EIR …`

6.4.2 accession

Description:	The protein's accession the peptide is associated with. In case no protein section is present in the file or the peptide was not assigned to a protein the field should be filled with “-“ or ”NA” respectively.
Type:	String

Example:	`PEH sequence accession …`
	`PEP KVPQVSTPTLVEVSR P02768 …`

6.4.3 unit_id

Description:	The unit the peptide was identified in.
Type:	String

Example:	`PEH sequence accession unit_id …`
	`PEP KVPQVSTPTLVEVSR P02768 PRIDE_1234 …`

6.4.4 unique

Description:	Indicates whether the peptide is unique for this protein in respect to the searched database.
Type:	Boolean

Example:	`PEH sequence accession unit_id unique …`
	`PEP KVPQVSTPTLVEVSR P02768 PRIDE_1234 0 …`
	`PEP VFDEFKPLVEEPQNLIK P02768 PRIDE_1234 1 …`

6.4.5 database

Description:	The protein database used for the search (could theoretically come from a different species) and the peptide sequence comes from.

Type:	String
Example:	PEH sequence accession unit_id unique database … PEP KVPQVSTPTLVEVSR P02768 PRIDE_1234 0 UniProtKB … PEP VFDEFKPLVEEPQNLIK P02768 PRIDE_1234 1 UniProtKB …

6.4.6 database_version

Description:	The protein database's version – in case there is no version available (custom build) the creation / download (e.g., for NCBI nr) date should be given.
Type:	String
Example:	PEH sequence accession unit_id unique database database_version … PEP KVPQVSTPTLVEVSR P02768 PRIDE_1234 0 UniProtKB 2011_11 … PEP VFDEFKPLVEEPQNLIK P02768 PRIDE_1234 1 UniProtKB 2011_11 …

6.4.7 search_engine

Description:	A "|" delimited list of search engine(s) used to identify this peptide. Search engines must be supplied as parameters.
Type:	Parameter List
Example:	PEH sequence … search_engine … PEP KVPQVSTPTLVEVSR … [MS,MS:1001207,Mascot,]|[MS,MS:1001208,Sequest,] … PEP VFDEFKPLVEEPQNLIK … [MS,MS:1001207,Mascot,] …

6.4.8 search_engine_score

Description:	A "|" delimited list of search engine score(s) for the given peptide. Scores SHOULD be reported using CV parameters whenever possible.
Type:	Parameter List
Example:	PRH sequence … search_engine_score … PEP KVPQVSTPTLVEVSR … [MS,MS:1001155,Sequest:xcorr,2] … PEP VFDEFKPLVEEPQNLIK … [MS,MS:1001171,Mascot score,47.2] …

6.4.9 reliability

Description:	The reliability of the given peptide identification. This must be supplied by the resource and has to be one of the following values: 1: high reliability 2: medium reliability 3: poor reliability Important: An identification's reliability is resource dependent.
Type:	Integer
Example:	PRH sequence … reliability … PEP KVPQVSTPTLVEVSR … 3 … PEP VFDEFKPLVEEPQNLIK … 1 …

6.4.10 modifications

Description:	The peptide's modifications. The position of the modification MUST be given relative to the peptide's start not the protein. For detailed information see protein table.
Type:	String
Example:	PRH sequence … modifications … PEP KVPQVSTPTLVEVSR … 10[0.8]-MOD:00412 …

	PEP VFDEFKPLVEEPQNLIK … NA …

6.4.11 retention_time

Description:	A comma-separated list of time points in seconds. Semantics may vary. This time should refer to the peptide's retention time if determined or the mid point between the first and last spectrum identifying the peptide.
Type:	Double List

Example:	`PRH sequence … retention_time …`
	`PEP KVPQVSTPTLVEVSR … 10.2 …`
	`PEP VFDEFKPLVEEPQNLIK … 15.8 …`

6.4.12 charge

Description:	The precursor's charge. In case multiple charge states for the same peptide are observed these should be reported as distinct entries in the peptide table. In case the charge is unknown "-" or "NA" should be used respectively.
Type:	Double

Example:	`PRH sequence … charge …`
	`PEP KVPQVSTPTLVEVSR … 2 …`
	`PEP VFDEFKPLVEEPQNLIK … 3 …`

6.4.13 mass_to_charge

Description:	The precursor's experimental mass to charge (m/z).
Type:	Double

Example:	`PRH sequence … mass_to_charge …`
	`PEP KVPQVSTPTLVEVSR … 1234.4 …`
	`PEP VFDEFKPLVEEPQNLIK … 123.4 …`

6.4.14 uri

Description:	A URI pointing to the peptide's entry in the experiment it was identified in (e.g., the peptide's PRIDE entry).
Type:	URI

Example:	`PRH sequence … uri …`
	`PEP KVPQVSTPTLVEVSR … http://www.ebi.ac.uk/pride/link/to/peptide …`
	`PEP VFDEFKPLVEEPQNLIK … http://www.ebi.ac.uk/pride/link/to/peptide …`

6.4.15 spectra_ref

Description:	Reference to a spectrum in a spectrum file. The reference must be in the format `ms_file[1-n]:{SPEC_REF}` where SPEC_REF MUST follow the format defined in mzIdentML. Multiple spectra MUST be referenced using a "|" delimited list.
Type:	String

Example:	`PRH sequence … spec_ref …`
	`PEP KVPQVSTPTLVEVSR … ms_file[1]:index=5 …`
	`PEP VFDEFKPLVEEPQNLIK … ms_file[2]:index=7|ms_file[2]:index=9 …`

6.4.16 peptide_abundance_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The peptide's abundance in the given subsample. This information can only be interpreted when identifying the subsample's properties through the

	experiment id + subsample number in the metadata section of the file.
Type:	Double

Example:	`PRH  sequence         …  peptide_abundance_sub[1]  …  peptide_abundance_sub[2] …`
	`PEP  KVPQVSTPTLVEVSR  …  0.4                       …  0.2                      …`

6.4.17 peptide_abundance_stdev_su b[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard deviation of the peptide's abundance. In case a peptide's abundance is given for a certain subsample the corresponding standard deviation column MUST also be present.
Type:	Double

Example:	`PRH  sequence         …  peptide_abundance_sub[1]  peptide_abundance_stdev_sub[1] …`
	`PEP  KVPQVSTPTLVEVSR  …  0.4                       0.2                            …`

6.4.18 peptide_abundance_std_error_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard error of the peptide's abundance. In case a peptide's abundance is given for a certain subsample the corresponding standard error column MUST also be present.
Type:	Double

Example:	`PRH  sequence        …  peptide_abundance_sub[1] … peptide_abundance_std_error_sub[1] …`
	`PEP  KVPQVSTPTLVEVSR …  0.4                      … 0.2                                …`

6.4.19 opt_*

Description:	**Optional** (this column MAY be present) Additional columns can be added to the end of the peptide table. These column headers MUST start with the prefix "opt_". Column names MUST only contain the following characters: 'A'-'Z', 'a'-'z', '0'-'9', '_', '-', '[', ']', and ':'. CV parameter accessions MAY be used for optional columns following the format: opt_cv_{accession}.
Type:	Column

Example:	`PRH  sequence        …  opt_my_value  opt_another_value`
	`PEP  KVPQVSTPTLVEVSR …  My value      some other value`

6.5 Small Molecule Section

The small molecule section is table-based. The small molecule section must always come after the metadata section, peptide section and or protein section if they are present in the file. All table columns MUST be tab-separated. There MUST NOT be any empty cells.
The columns in the small molecule section MUST be in the order they are presented in this document.

6.5.1 identifier

Description:	A list of "|" separated possible identifiers for these small molecules. The database identifier must be preceded by the resource description followed by a colon (in case this is not already part of the identifier format).
Type:	String List

Example:	`SMH  identifier     …`
	`SML  CID:00027395   …`

	SML HMDB:HMDB12345 …

6.5.2 unit_id

Description:	The unit the small molecule was identified in.
Type:	String
Example:	SMH identifier unit_id … SML CID:00027395 PRIDE_1234 … SML HMDB:HMDB12345 PRIDE_1234 …

6.5.3 chemical_formula

Description:	The chemical formula of the identified compound. This should be specified in Hill notation (EA Hill 1900), i.e. elements in the order C, H and then alphabetically all other elements. Counts of one may be omitted. Elements should be capitalized properly to avoid confusion (e.g., “CO” vs. “Co”). The chemical formula reported should refer to the neutral form. Charge state is reported by the charge field. This permits the comparison of positive and negative mode results. **Example:** N-acetylglucosamine would be encoded by the string “C8H15NO6”
Type:	String
Example:	SMH identifier unit_id chemical_formula … SML CID:00027395 PRIDE_1234 C17H20N4O2 …

6.5.4 description

Description:	The small molecule’s description / name.
Type:	String
Example:	SMH identifier … description … SML CID:00027395 … N-(2-phenylethyl)-3-[2-(pyridine-4-carbonyl)hydrazinyl]propanamide…

6.5.5 mass_to_charge

Description:	The small molecule’s precursor’s mass to charge ratio.
Type:	Double
Example:	SMH identifier unit_id … mass_to_charge … SML CID:00027395 PRIDE_1234 … 1234.5 …

6.5.6 charge

Description:	The precursor’s charge.
Type:	Double
Example:	SMH identifier unit_id … charge … SML CID:00027395 PRIDE_1234 … 2 …

6.5.7 retention_time

Description:	A comma-separated list of time points (in seconds). Semantics may vary. This time should refer to the small molecule’s retention time if determined or the mid point between the first and last spectrum identifying the small molecule.
Type:	Double List
Example:	SMH identifier unit_id … retention_time … SML CID:00027395 PRIDE_1234 … 10.2,11.5 …

6.5.8 taxid

Description:	The taxonomy id coming from the NEWT taxonomy for the species (if applicable).
Type:	Integer

Example:	`SMH  identifier     unit_id     …  taxid  …`
	`SML  CID:00027395   PRIDE_1234  …  NA     …`

6.5.9 species

Description:	The species as a human readable string (if applicable).
Type:	String

Example:	`SMH  identifier     unit_id     …  species  …`
	`SML  CID:00027395   PRIDE_1234  …  NA       …`

6.5.10 database

Description:	Generally references the used spectral library (if applicable).
Type:	String

Example:	`SMH  identifier     unit_id     …  database                 …`
	`SML  CID:00027395   PRIDE_1234  …  name of used database    …`

6.5.11 database_version

Description:	Either the version of the used database if available or otherwise the date of creation.
Type:	String

Example:	`SMH  identifier     unit_id     …  database_version   …`
	`SML  CID:00027395   PRIDE_1234  …  2011-12-22         …`

6.5.12 reliability

Description:	The reliability of the given small molecule identification. This must be supplied by the resource and has to be one of the following values: 1: high reliability 2: medium reliability 3: poor reliability Important: An identification's reliability is resource dependent.
Type:	Integer

Example:	`SMH  identifier     unit_id     …  reliability   …`
	`SML  CID:00027395   PRIDE_1234  …  3             …`

6.5.13 uri

Description:	A URI pointing to the small molecule's entry in the experiment it was identified in (e.g., the small molecule's PRIDE entry).
Type:	URI

Example:	`SMH  identifier     …  uri                                                 …`
	`SML  CID:00027395   …  http://www.ebi.ac.uk/pride/link/to/identification    …`

6.5.14 spectra_ref

Description:	Reference to a spectrum in a spectrum file. The reference must be in the format

	ms_file[1-n]:{SPEC_REF} where SPEC_REF must follow the format defined in mzIdentML. Multiple spectra can be referenced using a "|" delimited list.
Type:	String

Example:	`SMH  identifier    …  spectra_ref           …`
	`SML  CID:00027395  …  ms_file[1]:index=1002  …`

6.5.15 search_engine

Description:	A "|" delimited list of search engine(s) used to identify this small molecule. Search engines must be supplied as parameters.
Type:	Parameter List

Example:	`SMH  identifier    …  search_engine                   …`
	`SML  CID:00027395  …  [MS, MS:1001477, SpectraST,]  …`

6.5.16 search_engine_score

Description:	A "|" delimited list of search engine score(s) for the given small molecule. Scores SHOULD be reported using CV parameters whenever possible.
Type:	Parameter List

Example:	`SMH  identifier    …  search_engine_score                                       …`
	`SML  CID:00027395  …  [MS, MS:1001419, SpectraST:discriminant score F, 0.7]  …`

6.5.17 modifications

Description:	The small molecule's modifications. The position of the modification must be given relative to the small molecule's beginning. The exact semantics of this position depends on the type of small molecule identified. In case the position information is unknown or not applicable it should not be supplied. For detailed information see protein table.
Type:	String

Example:	`COM  example where an ammonium loss is found and the position is not`
	`COM  applicable in the given small molecule`
	
	`SMH  identifier    …  modifications  …`
	`SML  CID:00027395  …  CHEMMOD:+NH4   …`

6.5.18 smallmolecule_abundance_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The small molecule's abundance in the given subsample. This information can only be interpreted when identifying the subsample's properties through the experiment id + subsample number in the metadata section of the file.
Type:	Double

Example:	`SMH  identifier    …  smallmolecule_abundance_sub[1] …`
	`SML  CID:00027395  …  0.3                            …`

6.5.19 smallmolecule _abundance_stdev_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard deviation of the small molecule's abundance. In case the abundance for a respective subsample is given the standard deviation column MUST also be present.
Type:	Double

Example:	`SMH identifier … smallmolecule_abundance_sub[1] smallmolecule_abundance_stdev_sub[1]…`
	`SML CID:00027395… 0.3 0.04 …`

6.5.20 smallmolecule _abundance_std_error_sub[1-n] (Optional)

Description:	**Optional** (this column MAY be present) The standard error of the small molecule’s abundance. In case the abundance for a respective subsample is given the standard error column MUST also be present.
Type:	Double

Example:	`SMH identifier … smallmolecule_abundance_stderror_sub[1] …`
	`SML CID:00027395 … 0.04 …`

6.5.21 opt_*

Description:	**Optional** (this column MAY be present) Additional columns can be added to the end of the small molecule table. These column headers MUST start with the prefix “opt_”. Column names MUST only contain the following characters: ‘A’-‘Z’, ‘a’-‘z’, ‘0’-‘9’, ‘_’, ‘-’, ‘[’, ‘]’, and ‘:’. CV parameter accessions MAY be used for optional columns following the format: opt_cv_{accession}.
Type:	Column

Example:	`SMH identifier … opt_my_value opt_another_value`
	`SML CID:00027395 … My value some other value`

7. Conclusions

This document contains the specifications for using the mzTab format to represent results from peptide, small molecule and protein identification pipelines, in the context of a proteomics investigation. This specification constitutes a proposal for a standard from the Proteomics Standards Initiative. These artefacts are currently undergoing the PSI document process standardization process, which will result in a standard officially sanctioned by PSI.

8. Authors and Contributors

Johannes Griss, European Bioinformatics Institute
Timo Sachsenberg, University of Tübingen
Mathias Walzer, Center for Bioinformatics, University of Tübingen, Germany
Oliver Kohlbacher, Center for Bioinformatics, University of Tübingen, Germany
Henning Hermjakob, European Bioinformatics Institute
Juan Antonio Vizcaíno, European Bioinformatics Institute

Correspondence – Henning Hermjakob (hhe@ebi.ac.uk)

In addition to the authors, the following people contributed to the model development, gave feedback or tested mzTab:

- Andrew Jones, University of Liverpool
- Gerhard Thallinger, Graz University of Technology
- Jürgen Hartler, Graz University of Technology

- Martin Eisenacher, Medizinisches Proteom-Center, Ruhr-Universität Bochum
- Gerhard Mayer, Medizinisches Proteom-Center, Ruhr-Universität Bochum

9. References

- Bradner, S. (1997). Key words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force. RFC 2119.
- Martens, L., et al. (2011). "mzML--a community standard for mass spectrometry data." Mol Cell Proteomics 10(1): R110 000133.
- Jones, A. R., et al. (2012). "The mzIdentML data standard for mass spectrometry-based proteomics results." Mol Cell Proteomics doi:10.1074/mcp.M111.014381
- EA Hill (1900). "ON A SYSTEM OF INDEXING CHEMICAL LITERATURE; ADOPTED BY THE CLASSIFICATION DIVISION OF THE U. S. PATENT OFFICE." J. Am. Chem. Soc. 22 (8): 478–494. doi:10.1021/ja02046a005

10. Intellectual Property Statement

The PSI takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the PSI Chair.

The PSI invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to practice this recommendation. Please address the information to the PSI Chair (see contacts information at PSI website).

Copyright Notice

This document and the information contained herein is provided on an "AS IS" basis and THE PROTEOMICS STANDARDS INITIATIVE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."