The Twenty Minute Guide to mzQuantML

Andrew R. Jones[1], Juan Antonio Vizcaíno[2] and Gerhard Mayer[3]

[1] Institute of Integrative Biology, University of Liverpool, Liverpool L69 7ZB, UK
[2] EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD, UK
[3] Medizinisches Proteom-Center, Ruhr-Universität Bochum, Universitätsstr. 150, D-44801 Bochum, Germany

Abbreviations

CPTAC	Clinical Proteomic Technology Assessment for Cancer
CV	Controlled Vocabulary
Da	Dalton
iTRAQ	isobaric peptide Tags for Relative and Absolute Quantification
LC	Liquid Chromatography
MS	Mass Spectrometry
m/z	mass / charge ratio
HUPO-PSI	Human Proteome Organization - Proteomics Standards Initiative
MS1	PMF (= Peptide Mass Fingerprint)
MS2	Tandem Mass Spectrometry (PFF = Peptide Fragment Fingerprint)
MudPIT	Multi-Dimensional Protein Identification Technology
PSM	Peptide Spectrum Match
RT	Retention Time
SDS-PAGE	Sodium Dodecyl Sulfate – Polyacryl Amide Gel Electrophoresis
SILAC	Stable Isotope Labeling by Amino acid in Cell culture
SRM	Selected Reaction Monitoring
TMT	Tandem Mass Tag
XML	eXtensible Markup Language
XSD	XML Schema Definition

Introduction

The mzQuantML format from the Proteomics Standards Initiative (PSI) is designed to capture the output of quantitative software used in proteomics, based on mass spectrometry (MS). MS-based quantitation occurs through a variety of different methods, using stable isotopes labels (e.g. SILAC / N^{15}); by selected reaction monitoring (SRM), where the mass spectrometer is pre-configured to search for specific ions; chemical tags measured in the second stage of MS e.g. iTRAQ or TMT; or label free methods, based on comparing like-for-like peptide signals in parallel runs or counting the number of peptide-spectrum matches (PSMs) as a proxy measure of protein abundance. Software packages able to handle data from these different approaches have different ways of representing final results – typically relative or absolute abundance values about peptides or proteins, and

intermediate results and metadata. The mzQuantML format has been designed to act as a common output format from quantitative software packages used in proteomics, and to act as an input file for software packages able to post-process or visualise results. This document is intended as a simple guide of the most important features of the format, to help implementers understand the various structures and the expected encoding of different types of data.

The mzQuantML format is defined by an XML Schema Definition (XSD), a mapping file determining which controlled vocabulary (CV) terms must be used at certain points in files and a set of additional semantic validation rules that are checked by bespoke software. An instance file can be valid against the XSD but would not be a valid mzQuantML file unless it met the other defined rules. This design strategy was chosen so that a core XSD could be agreed which contains basic structures for representing quantitative values about protein groups, proteins, peptides and features (quantified regions in two-dimensional LC-MS space), but the specifications of how to encode a specific type of quantitative technique, such as SILAC, were met separately, thus allowing the core schema to remain stable, while proteomic technology continues to evolve.

The Core of mzQuantML

The core of the mzQuantML schema is illustrated graphically in Figure 1, corresponding to the way in which instance files can be constructed. At the top of a file in <CvList> (note: XML elements are enclosed in < >, XML attributes are presented in *italics*), in which the CVs used within the file must be specified. In most instances, this means simply referencing to the PSI-MS CV (http://psidev.cvs.sourceforge.net/viewvc/psidev/psi/psi-ms/mzML/controlledVocabulary/psi-ms.obo) and Unimod (1) or PSI-MOD (2), from which terms are used in the rest of the file. Next, the <Provider> of the instance document should be given, such as a person or organisation, with reference to contact details provided in <AuditCollection>.

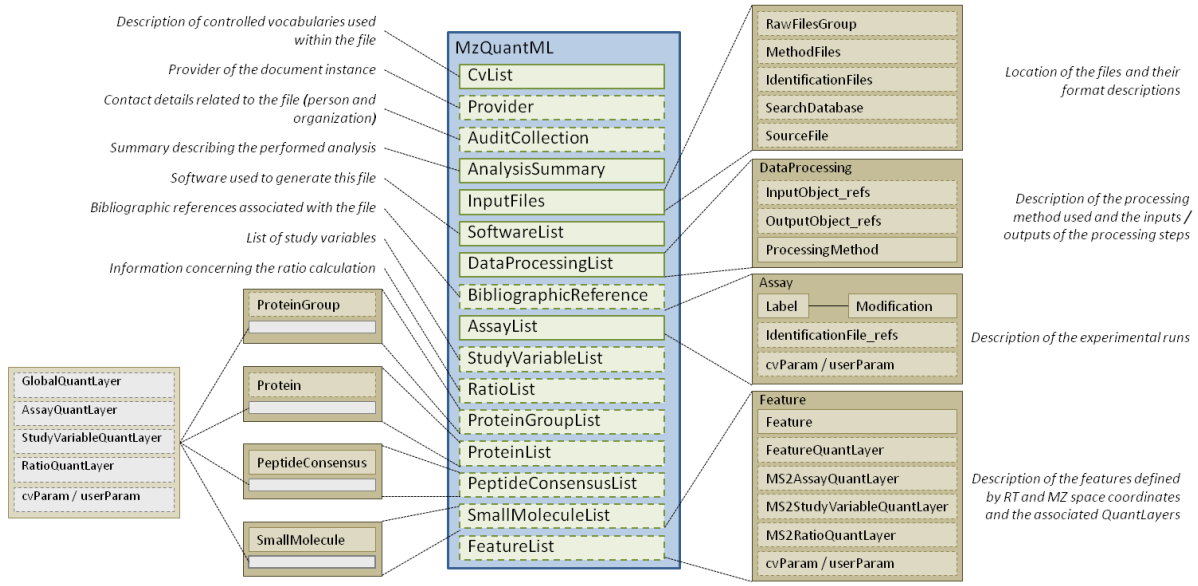

Figure 1 A graphical overview of the order of elements defined by the mzQuantML XSD.

AnalysisSummary

The next top-level element is <AnalysisSummary> which must contain specific CV terms, allowing processing software to interpret the type of file it is reading. In the first release of mzQuantML, the following terms are valid (and only one can be provided per file i.e. if two techniques have been employed they must be represented in separate files):

```
<cvParam accession="MS:1002018" cvRef="PSI-MS" name="MS1 label-based analysis"/>
<cvParam accession="MS:1001834" cvRef="PSI-MS" name="LC-MS label-free quantitation analysis"/>
<cvParam accession="MS:1002023" cvRef="PSI-MS" name="MS2 tag-based analysis"/>
<cvParam accession="MS:1001836" cvRef="PSI-MS" name="spectral counting quantitation
analysis"/>
```

Once processed by semantic validation software (see below), separate logic is applied to process the rest of the file.

InputFiles

The next part of metadata that must be represented is <InputFiles> (Figure 2), capturing the files processed by software to arrive at quantitative values. A file can contain zero to many instances of <RawFilesGroup>, where each <RawFilesGroup> represents a single unit of analysis on a mass spectrometer, holding one or more instances of <RawFile> - capturing for example a file location in the PSI's mzML format (3). In most cases, a <RawFilesGroup> will contain only one <RawFile>, but multiple files are allowed to handle cases where pre-fractionation has occurred e.g. by MudPIT or 1D SDS-PAGE and a set of raw files constitute a single MS run.

The optional element <MethodFiles> allows the specification of one-to-many instances of <MethodFile>, capturing separate parameter files essential for the quantitative analysis, such as input transitions for SRMs in the PSI's TraML format (4). <IdentificationFiles> stores one-to-many instances of <IdentificationFile> containing peptide or protein identifications output by a separate search engine where applicable, for example in the PSI's mzIdentML format (5). The file can contain zero to many instances of <SearchDatabase> describing, for example, the location of a protein sequence databases in FASTA format, from which quantified peptides/proteins were identified. Lastly, zero-to-many <SourceFile> elements can be specified to describe alternative format files that have been converted into the current instance of mzQuantML.

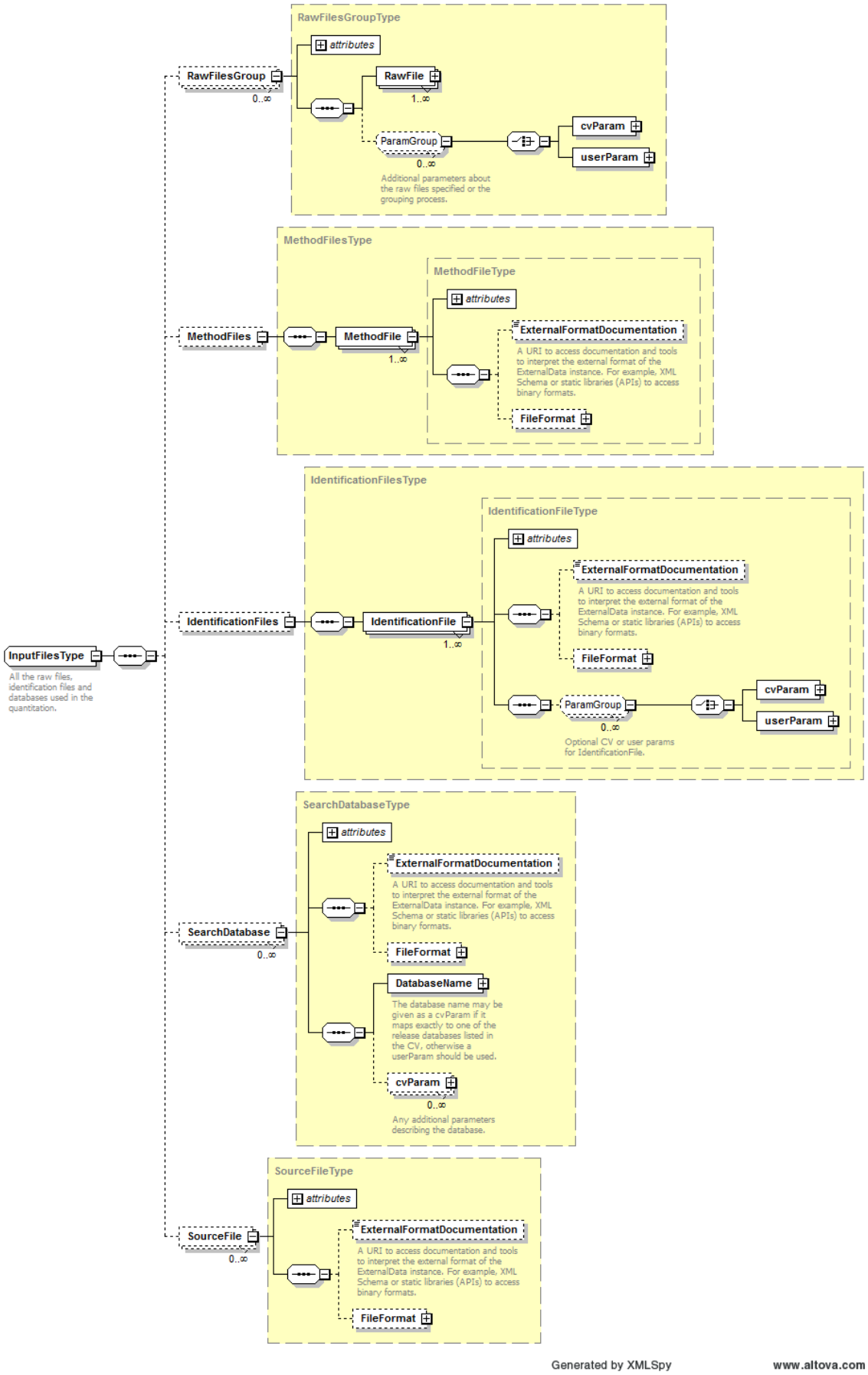

Figure 2 The <InputFiles> element of the mzQuantML XSD.

<SoftwareList>, <DataProcessingList> and <BibliographicReference>

The <SoftwareList> can contain one-to-many instances of <Software>, which captures a description of the software package(s) used for quantitation. The methods or processes employed by the software packages are captured in <DataProcessingList>, which can capture one-to-many instances of <DataProcessing> (Figure 3). <DataProcessing> describes one unit of an analysis pipeline performed with quantitative software (referenced under *software_Ref*). An *order* must be provided (integer) to demonstrate the order in which a particular analysis was performed within the entire pipeline. Optionally, references can be provided to mzQuantML data structures that were inputs or outputs of the process, such as a <FeatureList>, <PeptideConsensusList>, <ProteinList>, <ProteinGroupList> or a <QuantLayer> (all described in relevant sections below). One or more <ProcessingMethod> elements must be provided containing sub-steps within the analysis described in <DataProcessing> captured in <cvParam> or <userParam> (CV parameters or user parameters). Lastly, zero-to-many <BibliographicReference> elements may be provided to describe articles associated with the data captured in the file.

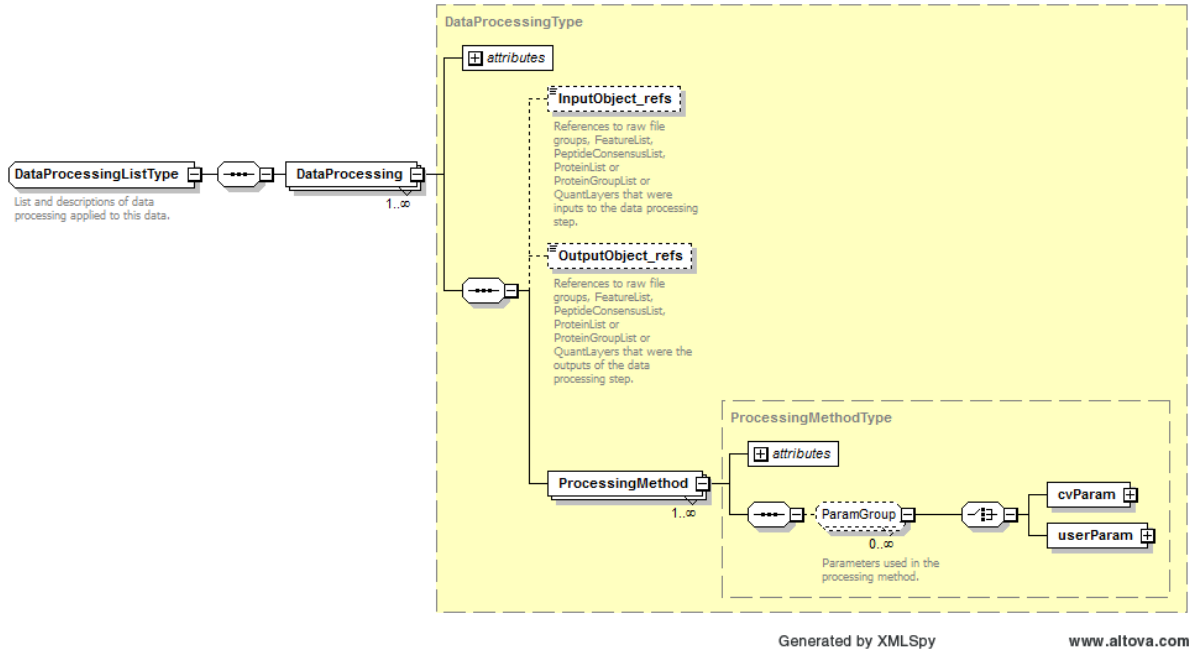

Figure 3 The <DataProcessing> element description from mzQuantML.

<AssayList>, <StudyVariableList> and <RatioList>

The <AssayList> and <StudyVariableList> elements capture descriptions of the experimental setup, which are referenced elsewhere in the file as the units about which data are reported. An <AssayList> must contain one or more <Assay> elements. Each <Assay> typically corresponds with a single biological sample that has been analysed (Figure 4). The <Assay> must have a unique identifier and may have a name, in which, for example, a human readable name for the sample could be provided. The <Assay> must reference to the <RawFilesGroup> to describe the raw MS data file(s) in which it has been analysed. As such, for label-free techniques in which different samples are analysed in parallel, there will typically be a one-to-one relationship from an <Assay> to a

<RawFilesGroup>. For multiplex techniques, such as a 4plex iTRAQ analysis, four instances of <Assay> would reference a single instance of <RawFilesGroup>. An <Assay> must specify a <Label> which differentiates it from other <Assay> instances within the same <RawFilesGroup>. The <Label> comprises a <Modification> element, for example importing a CV term from UniMod or PSI-MOD, and a mass shift. As an example, the four samples analysed by 4plex iTRAQ would each be represented by one <Assay> - each referencing the appropriate iTRAQ reagent used with that sample.

```
  <Assay rawFilesGroup_ref="raw1" name="114" id="_114">
    <Label>
    <Modification massDelta="145.0">
      <cvParam accession="MOD:01522" cvRef="PSI-MOD" value="114" name="iTRAQ4plex-114 reporter
fragment"/>
    </Modification>
    </Label>
  </Assay>
  <Assay rawFilesGroup_ref="raw1" name="115" id="_115">
    <Label>
    <Modification massDelta="145.0">
      <cvParam accession="MOD:01523" cvRef="PSI-MOD" value="115" name="iTRAQ4plex-115 reporter
fragment"/>
    </Modification>
    </Label>
  </Assay>
  ...
```

For a 2plex SILAC, experiment, two instances of <Assay> would be provided, in which the “light” sample must have a CV term “unlabeled sample” and the “heavy” sample would have a CV term containing the heavy SILAC reagent e.g. “Label:13C(6)15N(4)” (UNIMOD ID:267). In label-free analyses, all <Assay> elements must have a <Modification> element containing the “unlabeled sample” CV term.

```
<AssayList id="assaylist1">
  <Assay id="a_1452848987069318962" rawFilesGroup_ref="rfg_15787140749886983179">
    <Label>
      <Modification massDelta="0" >
        <cvParam cvRef="PSI-MOD" accession="MS:1002038" name="unlabeled sample" value="none"/>
      </Modification>
    </Label>
  </Assay>
  <Assay id="a_3151552639304580883" rawFilesGroup_ref="rfg_15787140749886983179">
    <Label>
      <Modification massDelta="8.0141988132" >
        <cvParam cvRef="PSI-MOD" accession="MOD:00582" name="6x(13)C,2x(15)N labeled L-lysine"
value="Lys8"/>
      </Modification>
      <Modification massDelta="10.0082686" >
        <cvParam  cvRef="PSI-MOD"  accession="MOD:00587"  name="6x(13)C,4x(15)N  labeled  L-
arginine" value="Arg10"/>
      </Modification>
    </Label>
  </Assay>
</AssayList>
```

<StudyVariableList> captures one-to-many instances of <StudyVariable> which references a set of <Assay> elements to act as a logical grouping, for example to describe a set of biological or technical replicates. The definition of what constitutes a <StudyVariable> is left up to the user of software to determine, but it is included to allow elements to be defined which can be referenced elsewhere and quantitative values can be attached to.

<RatioList> captures one-to-many instances of <Ratio>. A <Ratio> element references a numerator and a denominator, both of which must be instances of <Assay> or <StudyVariable>. A <Ratio> can be defined here, such that it can be referenced elsewhere in the file and quantitative values (i.e. ratios of <Assay> or <StudyVariable> elements) can be reported rather than single values only.

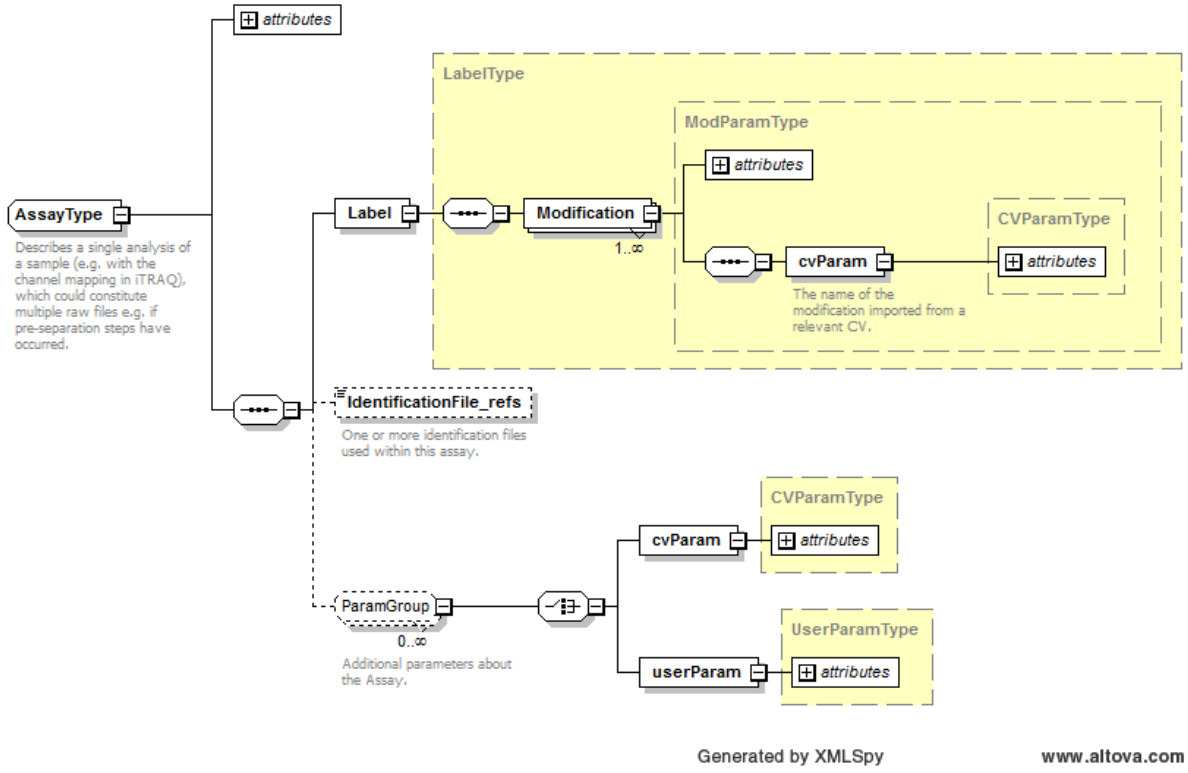

Figure 4 The <Assay> element in mzQuantML.

<ProteinGroupList>, <ProteinList>, <PeptideConsensusList> and <FeatureList>

All quantitative/abundance values are reported within <ProteinGroupList>, <ProteinList>, <PeptideConsensusList> and <FeatureList>, depending on whether the export software has produced values associated with protein groups, single proteins, peptides (matched across different samples analysed) or features (raw quantified values from MS prior to their assignment to peptide sequences), respectively.

The file can contain zero or one <ProteinGroupList> elements, which contains one-to-many instances of <ProteinGroup>. A <ProteinGroup> captures a set of references to <Protein> elements where one or more quantitative values are attached to a set of <Protein> elements rather than a single <Protein>, for example exported by software that chooses to solve the protein inference problem in this way.

The file can contain zero or one <ProteinList> elements, which contain one-to-many instances of <Protein>. Each <Protein> can have one-to-many references to <PeptideConsensus> elements, describing the peptides that were used to arrive at the quantitative values reported for the <Protein>.

The file can contain zero to many <PeptideConsensusList> elements, allowing software to export different sets of peptides and associated quantitative values as intermediates in a complex analysis pipeline. The <PeptideConsensusList> has a mandatory Boolean attribute finalResult and a valid file must contain only one <PeptideConsensusList> where 'finalResult = true' (checked by semantic validation software), allowing processing software to interpret the data set correctly. A <PeptideConsensus> element represents (and references) one or more features that report on the same peptide, for example across different assays. The process of matching features across different assays is dependent upon the type of technique (see specific examples for label-free, MS1 label-based and MS2 tag-based approaches below), but the same core structure can represent data about any of these techniques. A <PeptideConsensus> element can have a peptide sequence and a set of modifications defined by CV terms, although it is also possible for a <PeptideConsensus> element to exist without a sequence for techniques in which features are matched across samples and quantified, prior or without peptide identification.

The file can contain zero-to-many instances of <FeatureList>, which contains a mandatory reference to a <RawFilesGroup>. It is expected that there is one <FeatureList> per <RawFilesGroup> that is analysed. Each <FeatureList> contains one-to-many instances of <Feature>, which is defined as a quantified region in a 2D map of MS1 spectra (retention time (RT) versus mass over charge (m/z)). Each <Feature> must minimally have a value for retention time (set to null if liquid chromatography (LC) has not been performed or if the export software does not know the RT), m/z, charge and a unique identifier. An optional <MassTrace> may be provided to capture a set coordinates in RT and m/z space to define how the software has identified the isotope pattern of the feature in the 2D map.

The quantitative values for <ProteinGroup>, <Protein>, <PeptideConsensus> and <Feature> elements are provided in <QuantLayer> elements, described below.

QuantLayers

The primary purpose of mzQuantML is to provide the capability to communicate quantitative values attached to <ProteinGroup>, <Protein> and <PeptideConsensus> elements in an unambiguous manner, while ensuring that data files are not overly verbose (quantitative values assigned to <Feature> elements are handled separately below). The quantitative values assigned to each element type can be provided for each <Assay>, <StudyVariable> or <Ratio> specified in the file in 2D tables within <DataMatrix> elements, within <QuantLayers>. The type of quant layer defines the type of column of <DataMatrix> - available quant layers are <AssayQuantLayer>, <StudyVariableQuantLayer>, <RatioQuantLayer> or <GlobalQuantLayer> (described below). As such, an <AssayQuantLayer> within the <ProteinList> must contain a <DataMatrix> where the columns can only be references to <Assay> elements and each <Row> must reference to a <Protein> (Figure 5). With the exception of a <GlobalQuantLayer>, each <QuantLayer> can contain only one data type. If the software wishes to export multiple different types of quantitative values, say protein raw and normalised abundance, two <QuantLayers> are required. Semantic validation software has to check the following rules for the integrity of a <QuantLayer>:

- The <ColumnIndex> contains references to the correct type of object (<Assay>, <StudyVariable> or <Ratio>) as determined by the type of <QuantLayer>.
- The <Row> elements reference to the correct type of object as determined by the parent list (<ProteinGroupList>, <ProteinList>, <PeptideConsensusList>).
- The number of data values in each <Row> exactly matches the number of columns specified in the <ColumnIndex>. The specification document describes how null, “not a number” (NaN) or infinity values can be encoded, for example if a particular protein has not been measured in a given <Assay>.
- The <DataType> (for <AssayQuantLayer> and <StudyVariableQuantLayer>) is specified by a valid CV term according to the mapping file i.e. data types must be present in the PSI-MS CV. A <RatioQuantLayer> references to <Ratio> elements that describe the data type of the numerator and denominator separately.

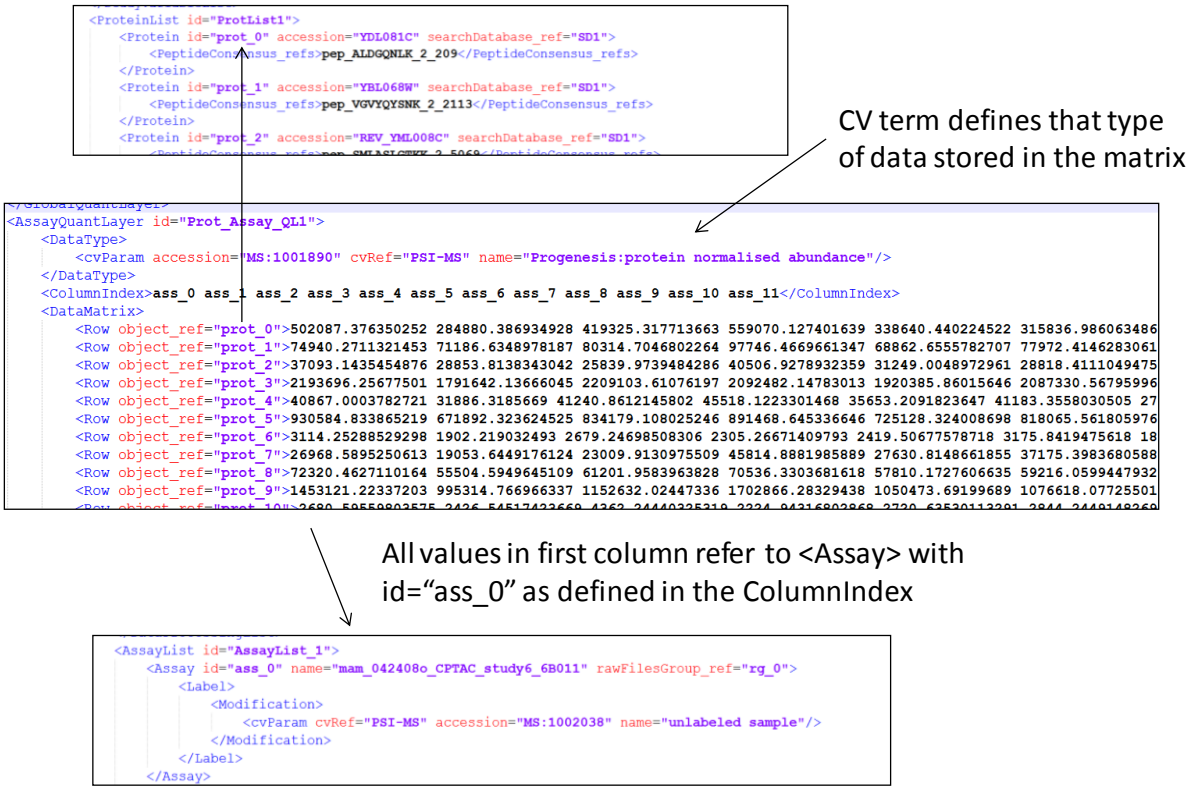

Figure 5 An <AssayQuantLayer> within the <ProteinList>. The <DataType> of the values within the <DataMatrix> is defined by a CV term. The <ColumnIndex> describes what columns of data are present in the <DataMatrix> via references to <Assay> elements. Each <Row> must reference to one <Protein> element. The number of data values in each <Row> must match exactly the number of columns defined in the <ColumnIndex>. In this example, 12 samples have been analysed by a label-free method. As such, the <ColumnIndex> contains references to 12 <Assay> elements, and each <Row> encodes 12 data values of type "Progenesis:protein normalised abundance".

A <GlobalQuantLayer> is a special type of <QuantLayer> in which the columns are <DataType> elements and the <Row> elements are defined by the parent list as before (references to <ProteinGroup>, <Protein> or <PeptideConsensus>). This provides the ability to represent values for a <Protein> say, which refers to the complete set of <Assay> elements analysed, such as global identification scores, p-values for detecting differential expression and so on.

A <FeatureList> can have only a limited subset of <QuantLayer> types: a <FeatureQuantLayer> (similar functionality to a <GlobalQuantLayer>) and a set of specific structures for techniques in which different samples (<Assay> elements) are compared within individual features i.e. MS2 chemical tag-based techniques, such as iTRAQ and TMT. The structures for handling these specific data types are called <MS2AssayQuantLayer>, <MS2StudyVariableQuantLayer> and <MS2RatioQuantLayer> (and are described in the iTRAQ / TMT section below). A <FeatureList> in general cannot have quant layers referring to <Assay> or <StudyVariable> since the feature list represents quantified regions in 2D RT versus m/z space, prior to matching processes in which multiple features are assigned as belonging to the same peptide (captured as <PeptideConsensus>).

Figure 6 A example of a mzQuantML file containing a <GlobalQuantLayer> within a <ProteinList>. The <ColumnDefintion> describes the <DataType> elements in each column of the <DataMatrix>. The number of values in each <Row> must match the number of <Column> elements.

How to model specific quantification approaches

1. MS1 intensity based Label-free

Certain software packages (such as 'Progenesis LC-MS'), align parallel MS runs in the retention time axis, and then quantify the identical feature region in all runs, even if there is no intensity in that region (thus giving a zero value). As such, the data can be represented in a regular fashion with an <AssayQuantLayer> or <StudyVariableQuantLayer> within <PeptideConsensus> elements referencing one feature for each assay. If the same peptide occurs in a different charge state or with a different modification, this is modelled by a different <PeptideConsensus> element. No attempt is made in the <PeptideConsensusList> to model the summed abundance of different features within the same assay that report on the same peptide. QuantLayers should be provided on the <PeptideConsensusList> and the <ProteinGroupList> or <ProteinList>. QuantLayers may be provided on the <FeatureList> for reporting additional data types about the features calculated prior to the feature matching process, although it is generally not recommend to report label-free data within <FeatureList> elements since this can produce a verbose encoding.

See example files:

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/label-free/CPTAC-Progenesis-small-example.mzq

http://mzquantml.googlecode.com/svn/trunk/examples/version1.0/label-free/maxquant-label-free.mzq

2. MS1 label-based

In MS1 label-based approaches two (or more) samples are mixed and analysed once by MS. In many approaches, pairs of features separated by a predictable mass shift are identified that report the relative abundance of the same peptide, from which a ratio can be calculated. To illustrate how mzQuantML encodes such an approach, the following example contains an encoding of a SILAC approach with a +8 Da shift for Lys and +10 Da shift for Arg, analysed in a single run, exported from the SILACAnalyzer tool within OpenMS.

See example files:

http://mzquantml.googlecode.com/svn/trunk/examples/version1.0/MS1Label/maxquant-silac.mzq

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/MS1Label/oms-data-silacanalyzer.mzq

There should be two <Assay> elements, each of which must refer to the same raw file(s) in <RawFilesGroup>. One of the <Assay> elements must contain details of the modification used to differentiate the peptide, such as the mass shift for the heavy Lys/Arg, the other <Assay> element must state that it is unmodified using the appropriate CV term.

If the data exporter wishes to communicate the full evidence trail, the primary results from analysis of each <RawFilesGroup> should be represented as a <FeatureList>. The <FeatureList> may contain data types about these features, including their raw intensity value. The next stage of the analysis is the finding of pairs of features that report on the same peptide (one of which has the expected mass shift). The result of this analysis should be captured by a <PeptideConsensusList> element, containing appropriate <AssayQuantLayer> or <RatioQuantLayer> elements for the data values that are exported.

The schema only allows one <ProteinGroupList> and/or one <ProteinList> and as such, if the exporter wishes to communicate values about proteins, the instance must contain one <ProteinGroupList> and/or one <ProteinList>, including the relevant QuantLayers for assays or study variables.

3. MS2 spectral counting

MS2 spectral counting approaches use the number of PSMs assigned to a given protein to estimate the protein abundance in the sample, following a variety of different normalisation schemes. As such, there is no requirement to model the intensity of MS1 features or provide QuantLayers on the <PeptideConsensusList>. A <PeptideConsensusList> may be provided to report the peptide sequences identified, although if accompanying mzIdentML files exist, it is not required. QuantLayers should only be provided on the <ProteinGroupList> and/or <ProteinList> to capture the abundance of proteins within assays or study variables.

See example files:

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/spectral-count/emPai_example_from_xTracker.mzq

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/spectral-count/mzQuantML_draft_spectralCount_from_Excel_MPC.mzq

4. MS2 tag-based

In MS2 tag-based approaches multiple samples are initially prepared in parallel and each set of peptides is modified by a chemical tag (such as an isobaric tag used in iTRAQ or TMT) by which it can be differentiated in MS2. The samples are mixed and analysed once by MS/MS, using the relative intensities of the tags when measured in MS2 to calculate the relative intensities of the tagged peptide in each of the source samples.

In this example, the use of 4plex iTRAQ with the tags 114, 115, 116 and 117 Da in a single run is described. The concept of a <Feature> describes a region of an MS1 spectrum/spectra and as such elements of <Feature> should be created to capture only the m/z and charge of the parent ion from which the iTRAQ intensities are calculated. Each <FeatureList> should contain at least one quant layer, of type <MS2AssayQuantLayer> or <MS2RatioQuantLayer> (where the ratios captured describe all assays e.g. 117/114, 116/114, 115/114) depending on what the data exporter wishes to communicate. The exporter may also include an <MS2StudyVariableQuantLayer>. A <QuantLayer> should only be attached to the <PeptideConsensusList> for approaches in which multiple different features reporting on the same peptide (such as sourced from different MS2 scans or different peptide charge states) are aggregated to provide a single value for the peptide.

See example files:

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/MS2Tag/iTraq_4plex_example_from_xTracker.mzq

http://code.google.com/p/mzquantml/source/browse/trunk/examples/version1.0/MS2Tag/oms-data-itraqanalyzer-id.mzq

Linkage from mzQuantML to mzIdentML and mzML

An mzQuantML instance may have been generated by software that used mzML for raw data input and mzIdentML for loading PSMs and/or proteins identified by a search engine. The link into an mzML file for raw data is provided via the file location on <RawFilesGroup> and m/z and RT coordinates on features identified from that <RawFilesGroup>.

```
<RawFilesGroup id="rg_0">
  <RawFile location="mam_042408o_CPTAC_study6_6B011.mzML" id="raw_0"/>
</RawFilesGroup>

<FeatureList id="Flist0" rawFilesGroup_ref="rg_0">
   <Feature charge="3" mz="398.871" rt="38.47" id="ft_0">
      <MassTrace>38.016 398.871 38.923 399.871</MassTrace>
   </Feature>
...
```

If multiple raw files are provided within a <RawFilesGroup> (where sample pre-fractionation has occurred), each <Feature> MUST be flagged with the <RawFile> from which it was derived:

```
<RawFilesGroup id="rg_0">
  <RawFile location="mam_042408o_CPTAC_study6_6B011_1.mzML" id="raw_0"/>
  <RawFile location="mam_042408o_CPTAC_study6_6B011_2.mzML" id="raw_1"/>
</RawFilesGroup>

<FeatureList id="Flist0" rawFilesGroup_ref="rg_0">
   <Feature charge="3" mz="398.871" rt="38.47" id="ft_0" rawFile_ref= "raw_0" />
   <Feature charge="2" mz="647.296" rt="50.12" id="ft_1" rawFile_ref= "raw_1" />

...
```

For associating an mzQuantML instance to identification files in mzIdentML, the following elements are used.

```
<IdentificationFiles>
  <IdentificationFile searchDatabase_ref="SD1" location="file://idents.mzid" id="idfile_1"/>
</IdentificationFiles>


<PeptideConsensus searchDatabase_ref="SD1" charge="2" id="pep_GAPEIDVLEGETDTK_2_21711">
   <PeptideSequence>GAPEIDVLEGETDTK</PeptideSequence>
   <EvidenceRef  feature_ref="ft_216"  identificationFile_ref="idfile_1"  id_refs="SII_69413_1"
assay_refs="ass_0"/>
   <EvidenceRef  feature_ref="ft_217"  identificationFile_ref="idfile_1"  id_refs="SII_69415_1"
assay_refs="ass_1"/>
```

A <PeptideConsensus> element can reference objects within one or more mzIdentML files using <EvidenceRef>. <EvidenceRef> can store the unique ID of a <SpectrumIdentificationItem> element e.g. SII_69413_1, and the filename (via <IdentificationFile>). The <EvidenceRef> object also references the corresponding <Feature> to which the identification has been mapped and the <Assay> to which the <Feature> belongs (e.g. in label-free) or has been assigned (e.g. in SILAC after pair identification).

Outstanding issues and more advanced features

This guide only describes the basic features of mzQuantML for supporting the four techniques included in the initial release. Below is a list of known issues and areas in which future updates to this document will be released.

- The development of semantic validation rules for SRM-based approaches is on-going and we welcome further input and examples in this area.
- The schema contains a <SmallMoleculeList> element for use in metabolomics, but the semantic validation rules and appropriate examples are yet to be developed.

References

[1] D.M. Creasy, J.S. Cottrell, Unimod: Protein modifications for mass spectrometry, Proteomics, 4 (2004) 1534-1536.
[2] L. Montecchi-Palazzi, R. Beavis, P.A. Binz, R.J. Chalkley, J. Cottrell, D. Creasy, J. Shofstahl, S.L. Seymour, J.S. Garavelli, The PSI-MOD community standard for representation of protein modification data, Nat Biotechnol, 26 (2008) 864-866.
[3] L. Martens, M. Chambers, M. Sturm, D. Kessner, F. Levander, J. Shofstahl, W.H. Tang, A. Rompp, S. Neumann, A.D. Pizarro, L. Montecchi-Palazzi, N. Tasman, M. Coleman, F. Reisinger, P. Souda, H. Hermjakob, P.A. Binz, E.W. Deutsch, mzML--a community standard for mass spectrometry data, Mol Cell Proteomics, 10 (2011) R110 000133.
[4] E.W. Deutsch, M. Chambers, S. Neumann, F. Levander, P.A. Binz, J. Shofstahl, D.S. Campbell, L. Mendoza, D. Ovelleiro, K. Helsens, L. Martens, R. Aebersold, R.L. Moritz, M.Y. Brusniak, TraML--a standard format for exchange of selected reaction monitoring transition lists, Mol Cell Proteomics, 11 (2012) R111 015040.
[5] A.R. Jones, M. Eisenacher, G. Mayer, O. Kohlbacher, J. Siepen, S.J. Hubbard, J.N. Selley, B.C. Searle, J. Shofstahl, S.L. Seymour, R. Julian, P.A. Binz, E.W. Deutsch, H. Hermjakob, F. Reisinger, J. Griss, J.A. Vizcaino, M. Chambers, A. Pizarro, D. Creasy, The mzIdentML data standard for mass spectrometry-based proteomics results, Mol Cell Proteomics, 11 (2012) M111 014381.