nmrML - documentary material

nmrML: an open standard for the description, storage, and exchange of NMR data

This document provides a more practical and extensive background information on the set-up and the usage of the nmrML data standard for open access NMR data. It also provides deeper insights into the exact mappings of vendor format parameters to nmrML elements and nmrCV terms, as used by the nmrML converter. The detailed information provided here amends the nmrML paper in MCP.

Keep in mind that the more updated examples ARE ALWAYS FOUND ON OUR WEBSITE at http://nmrml.org/examples/

The Github pages are available under https://github.com/nmrML/nmrML

Content

Birdseye view on nmrML and overall set-up

A bird's eye view of nmrML and its major use cases is provided in Figure 1.

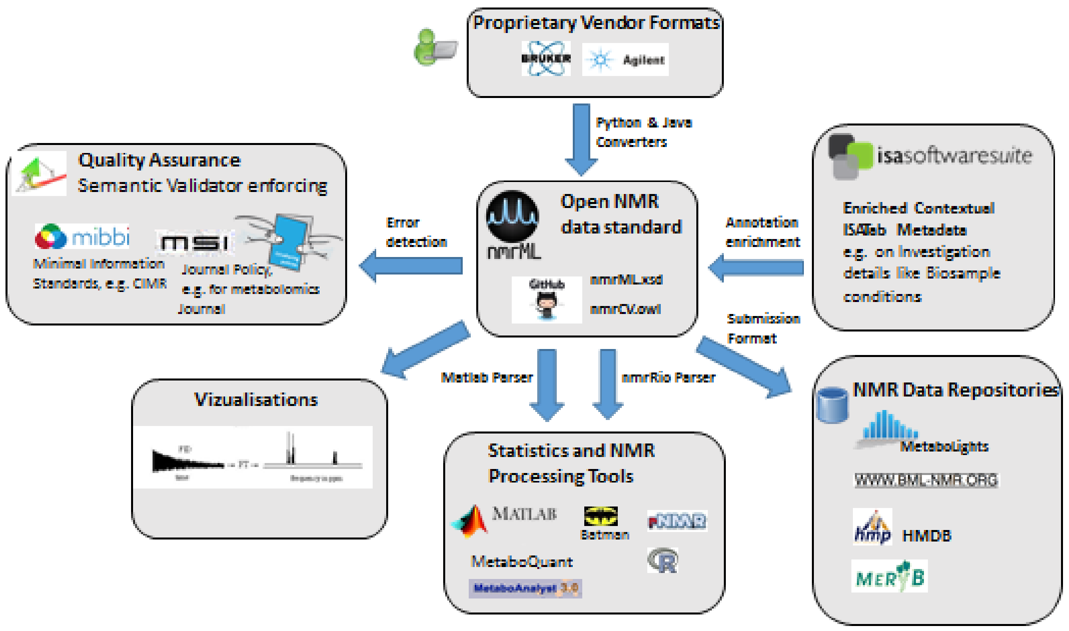


Figure 1: Illustration of NMR data management facilitation by means of nmrML. Primary advantages are the support of standardised repositories, and processing tools also in Workflow Systems,i.e. via available parsers and format converters (arrows). Secondary use cases are granular NMR data retrieval, comparison, visualization, as well as quality assurance and richer standardized metadata annotations i.e. via ontologies. IsaTab is a metadata annotation format and tool suite to support tabular capture of the broader high-level Experimentation and Study metadata.

The nmrML set-up (see Fig. 2) leverages on several previous efforts: The Proteomics Standards initiative (PSI) has developed XML based data exchange standards for mass spectrometry based proteomics (mzML and its accompanying mzML CV). Their mzML standard[1] proved of great usability in proteomics data standardization and was selected as a role model, guiding our basic design decisions regarding nmrMLs structural set-up.

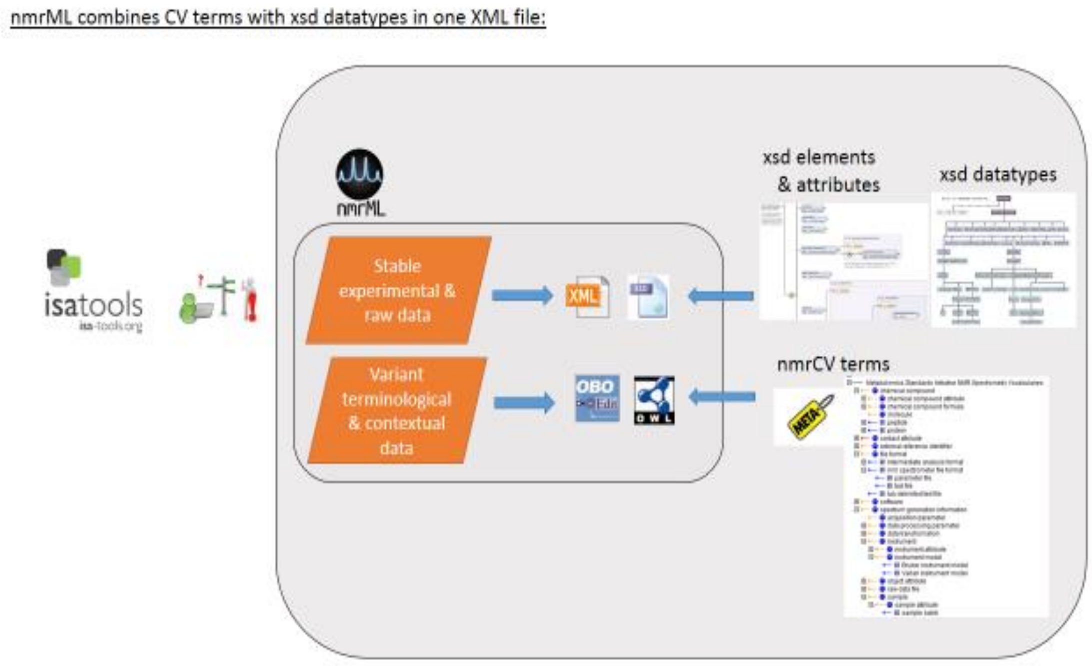

nmrML combines CV terms with xsd datatypes in one XML file:
nmrML
isatools
isa-tools.org
Stable experimental & raw data
Variant terminological & contextual data
XML
OBO Edit
OWL
xsd elements & attributes
xsd datatypes
nmrCV terms
META

Figure 2: nmrML consists of an XSD specification, capturing raw data in an XML file, and a controlled vocabulary (CV), capturing standardized descriptors as CV terms in the XML file. CV terms capture the more variant contextual NMR terminology, and are provided by external CVs. CV term usage allows the concise description of an NMR instrument configuration by means of a series of CV terms for probe heads used, auto-samplers, brands, models, etc. ISA can be populated from nmrML files via the nmrML2ISA tool and allows further validation layers.

nmrMLs major elements: Overview on its top level XSD Structure

nmrML consists of the following top level XSD structure elements (Fig. 3).

[1] Martens,L., Chambers,M., Sturm,M. et al. (2011) mzML—a community standard for mass spectrometry data. Mol. Cell Proteomics, 10, R110000133. http://www.ncbi.nlm.nih.gov/pubmed/20716697

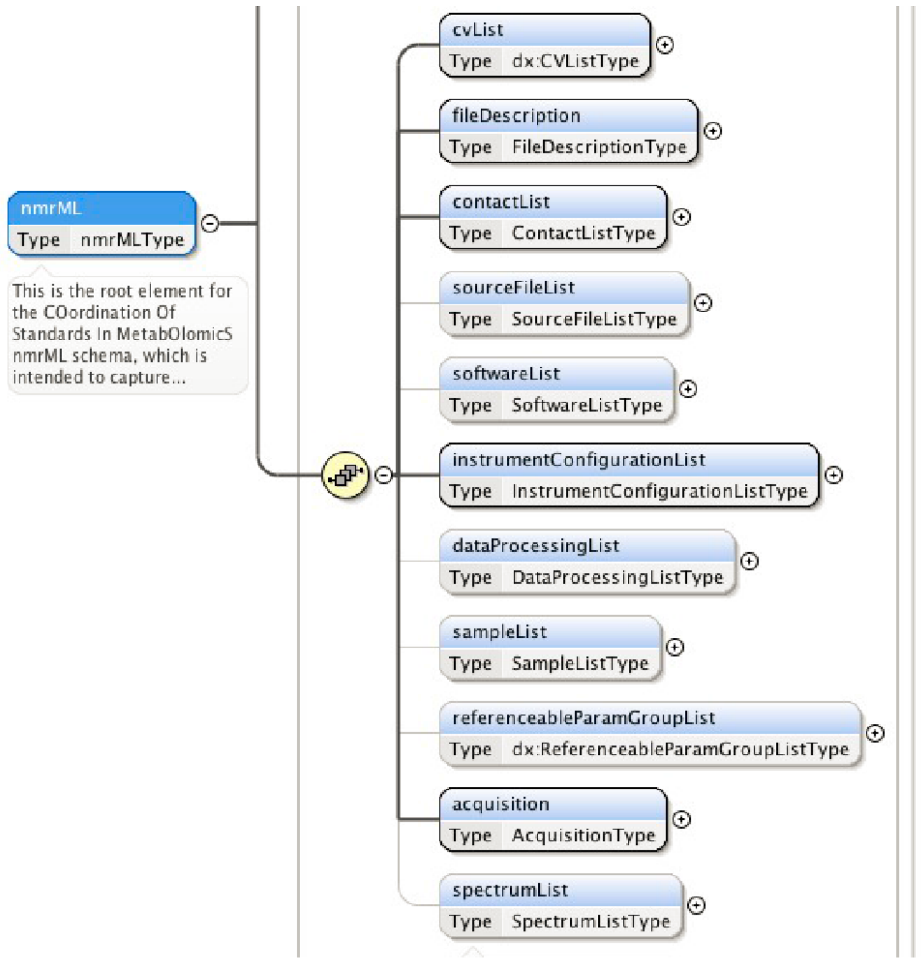


Figure 3: The top level XML elements of the nmrML.xsd, illustrating its main elements and order thereof (Oxygen Editors Design view screenshot).

We now describe the **basic top level elements** occurring in nmrML, as appearing in Fig. 3:

'**cvList**' defines the controlled vocabularies from which terms are drawn in the XML data file. This allows tools to look-up additional CV metadata, and even the complete taxonomic (is_a) hierarchy of a term. Traversing root-wards in a taxonomy is called abstraction, traversing leaf-wards is called specialisation or subsumption.

'**fileDescription**' allows the CV based description of the nmrML file itself which allows for easy categorization of different types of nmrML instances e.g. 1D vs. 2D NMR data sets. This is expected to aid processing software in skipping files that do not contain appropriate spectrum types for it.

'**contactLis**t' captures information that allows one to contact the original creators/corresponding authors of a file/paper in the case that further clarification is needed.

'**sourceFileLis**t' captures attributes like name, ID, location and CV descriptors for the original NMR raw data files that were used to generate the nrmML file. This includes files that were required during the acquisition of the spectrum, for example a Varian processing parameter file or a source code file for a pulse program

‘**softwareLis**t’ captures references to software that was used during data acquisition and processing, and may include several different pieces of software.

‘**informationConfigurationList**’ contains information about the configuration of an instrument beyond the acquisition parameters, for example the brand and model of the instrument.

‘**dataProcessingList**’ captures the list and order of software applied together with their respective processing methods by means of CV based descriptions.

‘**sampleList**’ captures lists of NMR samples by means of their solvent properties, buffer, pH value, chemical shift and concentration standards used.

‘**referenceableParamGroupList**’ allows to capture a multiplicity of CV descriptions to be identified and re-used again throughout an nmrML file to avoid redundancy.

‘**acquisition**’ captures the processing parameters used during the acquisition (see below). Since vendors have their own set of names for each of these parameters, we have standardized them with hopefully intuitive clear names. This element also contains the captured FID raw data.

‘**spectrumList**’ contains one or more spectra in the frequency domain.

For further documentation, we refer to the example instantiation (below and on nmrML.org), to the HTML documentation of the XSD, and the XSD itself, in which extensive element annotations explain the scope and usage of all nmrML elements.

Description of NMR acquisition parameters in nmrML

To see what is captured under nmrML/acquisition/acquisition1D/acquisitionParameterSet/ look at Fig. 4:

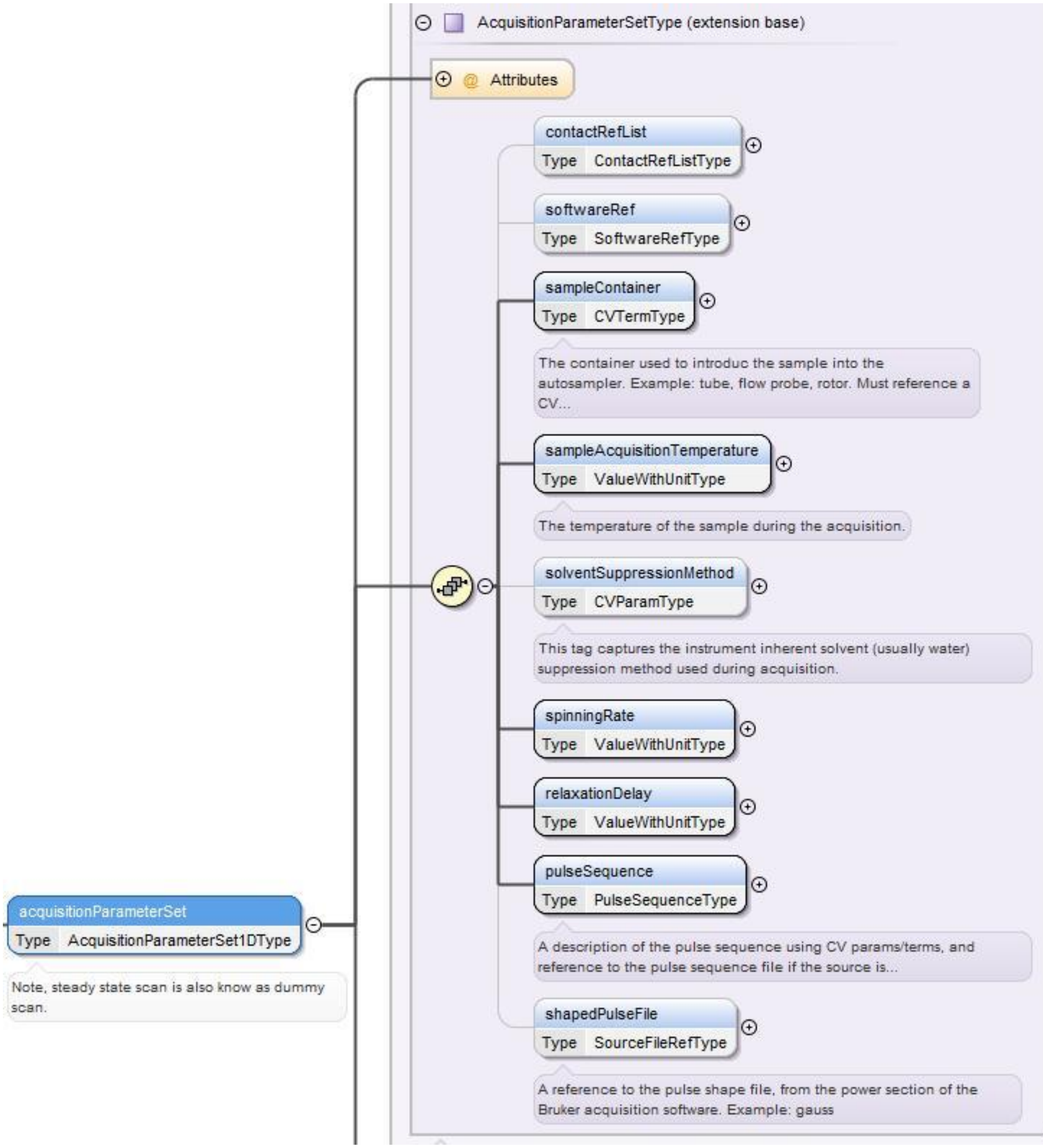

AcquisitionParameterSetType (extension base)
Attributes
contactRefList
Type ContactRefListType
softwareRef
Type SoftwareRefType
sampleContainer
Type CVTermType
The container used to introduc the sample into the autosampler. Example: tube, flow probe, rotor. Must reference a CV...
sampleAcquisitionTemperature
Type ValueWithUnitType
The temperature of the sample during the acquisition.
solventSuppressionMethod
Type CVParamType
This tag captures the instrument inherent solvent (usually water) suppression method used during acquisition.
spinningRate
Type ValueWithUnitType
relaxationDelay
Type ValueWithUnitType
pulseSequence
Type PulseSequenceType
A description of the pulse sequence using CV params/terms, and reference to the pulse sequence file if the source is...
shapedPulseFile
Type SourceFileRefType
A reference to the pulse shape file, from the power section of the Bruker acquisition software. Example: gauss
acquisitionParameterSet
Type AcquisitionParameterSet1DType
Note, steady state scan is also know as dummy scan.

Figure 4. An example section from the nmrML XSD is shown with the elements required to capture NMR acquisition parameters for 1D experiments.

As a more concrete example for an XSD element specification, we will now look at the sourceFile element (Fig. 5) which is quite complex as its XML instance is to be specified via XML attributes together with constituted CV references, i.e. taken from the unrestrictive ParamGroupType. sourceFile instances have to have basic attributes like name and location URI, but can have additional CV reference descriptions specified via the ParamGroupType extension base. The latter allows a CV based description using any of the basic CV term reference types seen in Table 2 (below).

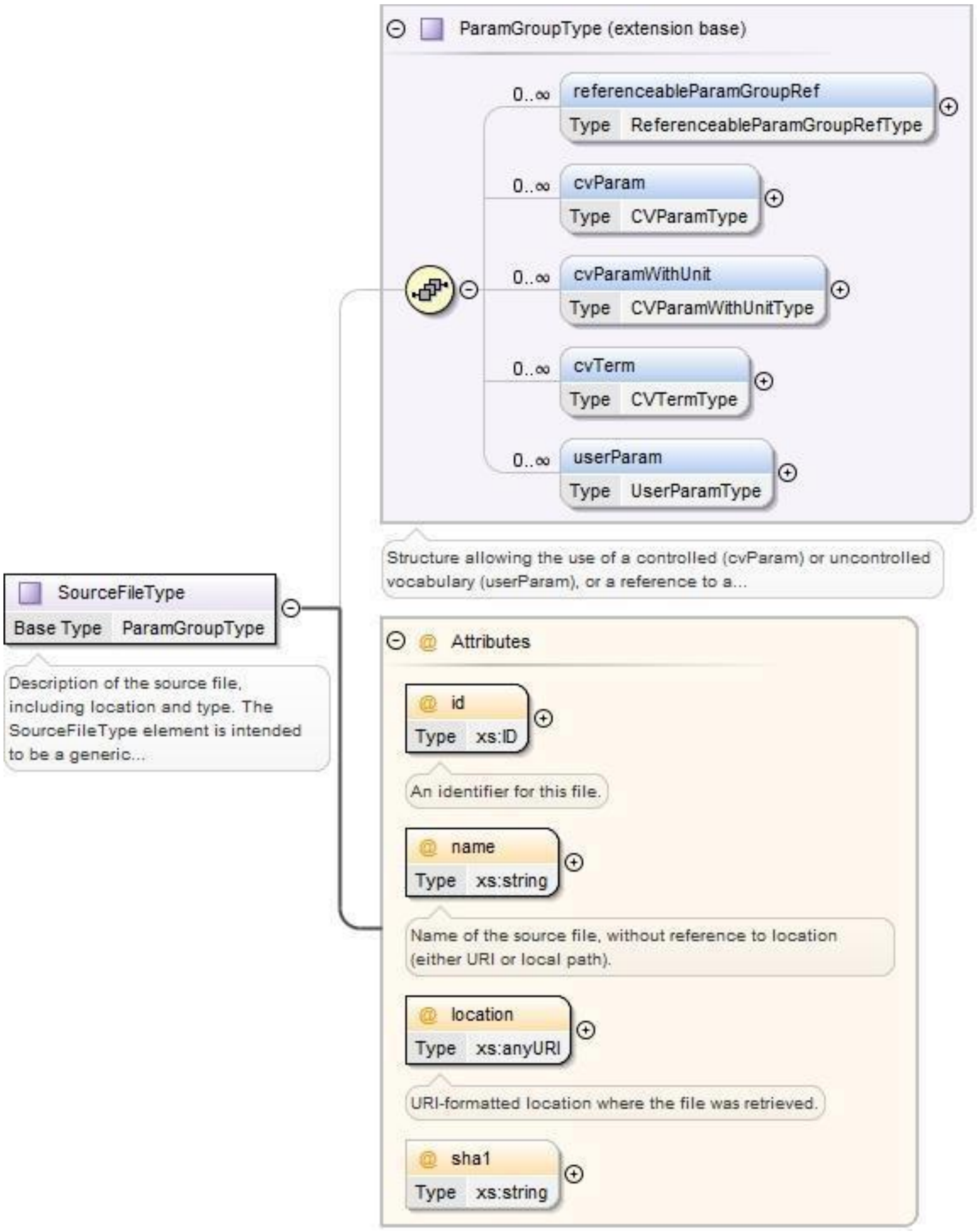

ParamGroupType (extension base)
0..∞
referenceableParamGroupRef
Type ReferenceableParamGroupRefType
0..∞
cvParam
Type CVParamType
0..∞
cvParamWithUnit
Type CVParamWithUnitType
0..∞
cvTerm
Type CVTermType
0..∞
userParam
Type UserParamType
Structure allowing the use of a controlled (cvParam) or uncontrolled vocabulary (userParam), or a reference to a...
SourceFileType
Base Type ParamGroupType
Description of the source file, including location and type. The SourceFileType element is intended to be a generic...
Attributes
id
Type xs:ID
An identifier for this file.
name
Type xs:string
Name of the source file, without reference to location (either URI or local path).
location
Type xs:anyURI
URI-formatted location where the file was retrieved.
sha1
Type xs:string

Figure 5: Specification of an unrestrictive CV term usage for the nmrML SourceFileType, allowing all possible CVterm usage modes. The XSD code can be seen in Fig. 7 below. An example XML instance can be found below in Fig. 8 below.

Processed data in nmrM

We have added the capability to also store processing outcomes in the nmrML format.

We have added two elements in the XSD :

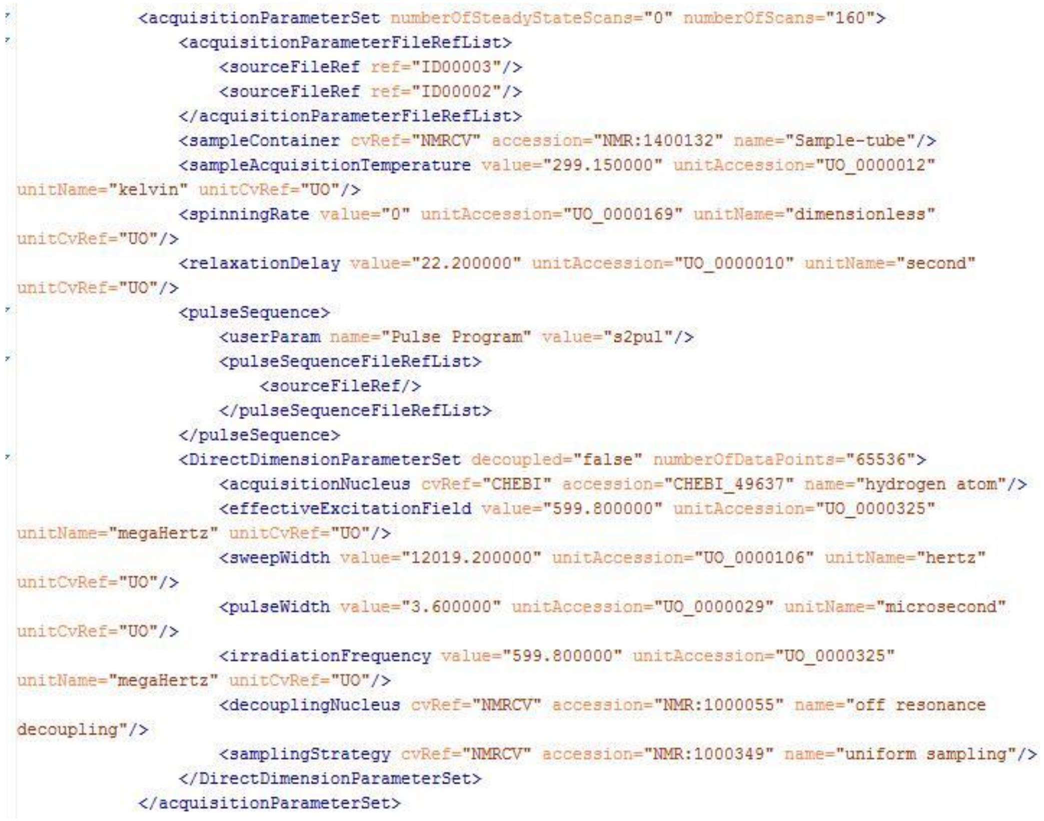
1/ the "acquisitionParameterFileRefList" element to the acquisitionParameterSet Type

2/ the "processingParameterFileRefList" element to the Spectrum1D Type

For the time being, they are declared as optional these two elements (i.e. minoccurs=0) so that the already examples online will still validated. (see https://github.com/nmrML/nmrML/issues/171)

nrmML data examples

Fig. 6 provides an example instantiation for an acquisition XSD specification, to explain the actual practical usage of nmrML for data capture.



```
            <acquisitionParameterSet numberOfSteadyStateScans="0" numberOfScans="160">
                <acquisitionParameterFileRefList>
                    <sourceFileRef ref="ID00003"/>
                    <sourceFileRef ref="ID00002"/>
                </acquisitionParameterFileRefList>
                <sampleContainer cvRef="NMRCV" accession="NMR:1400132" name="Sample-tube"/>
                <sampleAcquisitionTemperature value="299.150000" unitAccession="UO_0000012"
unitName="kelvin" unitCvRef="UO"/>
                <spinningRate value="0" unitAccession="UO_0000169" unitName="dimensionless"
unitCvRef="UO"/>
                <relaxationDelay value="22.200000" unitAccession="UO_0000010" unitName="second"
unitCvRef="UO"/>
                <pulseSequence>
                    <userParam name="Pulse Program" value="s2pul"/>
                    <pulseSequenceFileRefList>
                        <sourceFileRef/>
                    </pulseSequenceFileRefList>
                </pulseSequence>
                <DirectDimensionParameterSet decoupled="false" numberOfDataPoints="65536">
                    <acquisitionNucleus cvRef="CHEBI" accession="CHEBI_49637" name="hydrogen atom"/>
                    <effectiveExcitationField value="599.800000" unitAccession="UO_0000325"
unitName="megaHertz" unitCvRef="UO"/>
                    <sweepWidth value="12019.200000" unitAccession="UO_0000106" unitName="hertz"
unitCvRef="UO"/>
                    <pulseWidth value="3.600000" unitAccession="UO_0000029" unitName="microsecond"
unitCvRef="UO"/>
                    <irradiationFrequency value="599.800000" unitAccession="UO_0000325"
unitName="megaHertz" unitCvRef="UO"/>
                    <decouplingNucleus cvRef="NMRCV" accession="NMR:1000055" name="off resonance
decoupling"/>
                    <samplingStrategy cvRef="NMRCV" accession="NMR:1000349" name="uniform sampling"/>
                </DirectDimensionParameterSet>
            </acquisitionParameterSet>
```

Figure 6. Example nrmML xml code instantiating the acquisition parameters for an 1D NMR experiment. This is how your data look like, when converted into nmrML. Another example with more details provides is shown in Fig 8.

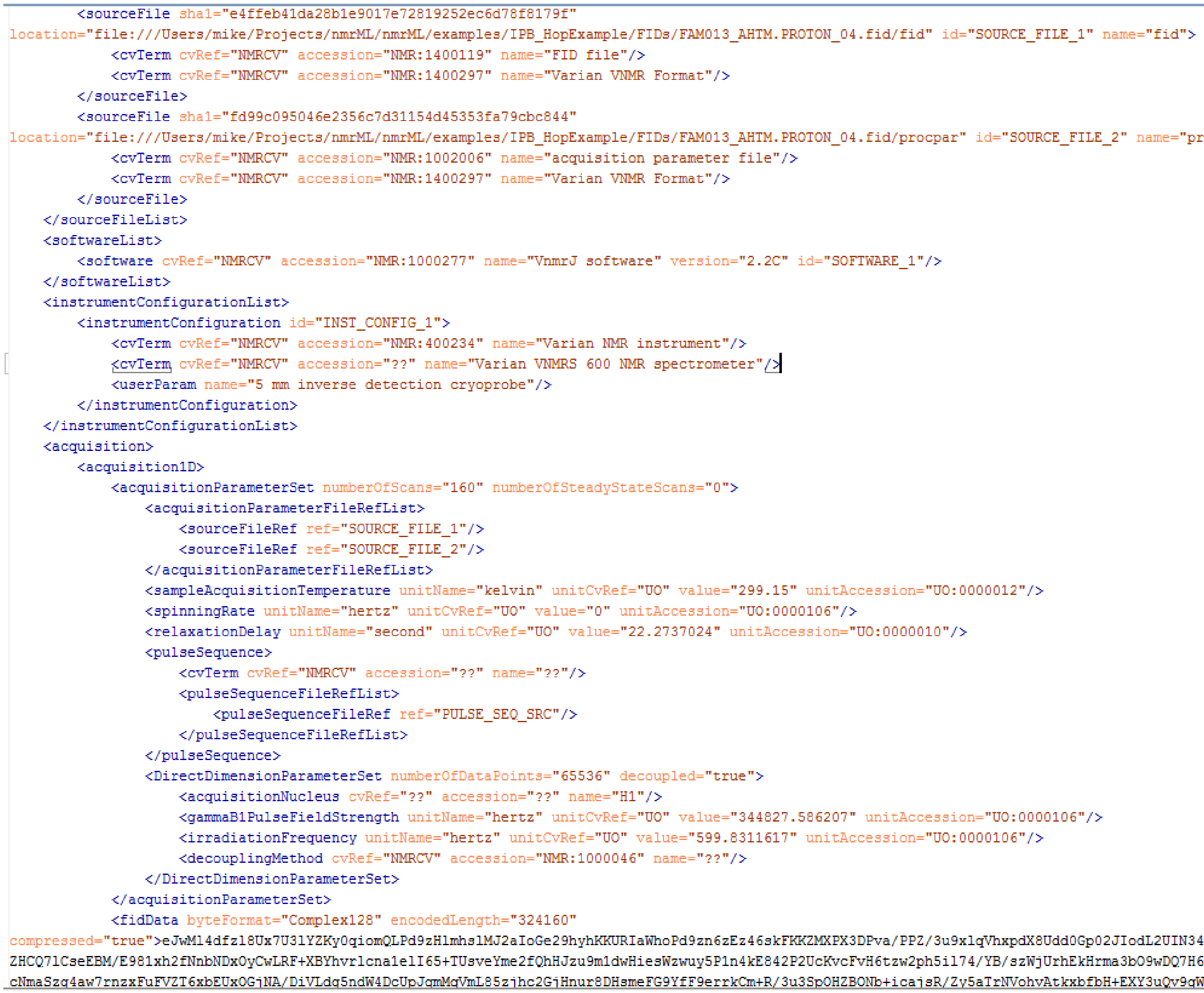
```
<xs:complexType name="SourceFileType">
  <xs:annotation> <xs:documentation>Description of the source file, including location and type.</xs:documentation>  </xs:annotation>
  <xs:complexContent mixed="false">
```

```
   <xs:extension base="ParamGroupType">
    <xs:attribute name="id" type="xs:ID" use="required">
     <xs:annotation> <xs:documentation>An identifier for this file.</xs:documentation> </xs:annotation>
    </xs:attribute> <xs:attribute name="name" type="xs:string" use="required"> <xs:annotation>
      <xs:documentation>Name of the source file, without reference to location (either URI or
       local path).</xs:documentation> </xs:annotation> </xs:attribute>
    <xs:attribute name="location" type="xs:anyURI" use="required">
     <xs:annotation> <xs:documentation>URI-formatted location where the file was          retrieved.</xs:documentation> </xs:annotation>
    </xs:attribute>
    <xs:attribute name="sha1" type="xs:string"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="SourceFileRefType">
  <xs:attribute name="ref" type="xs:IDREF" use="required">
   <xs:annotation> <xs:documentation>This attribute must reference the 'id' of the appropriate
     sourceFile.</xs:documentation> </xs:annotation>
  </xs:attribute>
 </xs:complexType>
```

Fig. 7 : nmrML XSD section specifying valid SourceFile descriptions in nmrML. A graphic representation of this XSD section is found in Figure 5.

In the following we explain the nmrML usage in an example that represents a typical metabolomics experimental set-up: Thirteen hop plant ecotypes were profiled for interesting secondary metabolites using MS and NMR in combination[2]. Figure 8 illustrates how 1D acquisition and raw FID data is stored in an nmrML xml instance for one of the hop variants (AHTM).

[2]Farag, M., Porzel, A., Schmidt, J. & Wessjohann, L. Metabolite profiling and fingerprinting of commercial cultivars of *Humulus lupulus* L. (hop) - a comparison of MS and NMR methods in metabolomics Metabolomics 8, 492-507, (2012)



```xml
        <sourceFile sha1="e4ffeb41da28b1e9017e72819252ec6d78f8179f"
location="file:///Users/mike/Projects/nmrML/nmrML/examples/IPB_HopExample/FIDs/FAM013_AHTM.PROTON_04.fid/fid" id="SOURCE_FILE_1" name="fid">
            <cvTerm cvRef="NMRCV" accession="NMR:1400119" name="FID file"/>
            <cvTerm cvRef="NMRCV" accession="NMR:1400297" name="Varian VNMR Format"/>
        </sourceFile>
        <sourceFile sha1="fd99c095046e2356c7d31154d45353fa79cbc844"
location="file:///Users/mike/Projects/nmrML/nmrML/examples/IPB_HopExample/FIDs/FAM013_AHTM.PROTON_04.fid/procpar" id="SOURCE_FILE_2" name="pr
            <cvTerm cvRef="NMRCV" accession="NMR:1002006" name="acquisition parameter file"/>
            <cvTerm cvRef="NMRCV" accession="NMR:1400297" name="Varian VNMR Format"/>
        </sourceFile>
    </sourceFileList>
    <softwareList>
        <software cvRef="NMRCV" accession="NMR:1000277" name="VnmrJ software" version="2.2C" id="SOFTWARE_1"/>
    </softwareList>
    <instrumentConfigurationList>
        <instrumentConfiguration id="INST_CONFIG_1">
            <cvTerm cvRef="NMRCV" accession="NMR:400234" name="Varian NMR instrument"/>
            <cvTerm cvRef="NMRCV" accession="??" name="Varian VNMRS 600 NMR spectrometer"/>
            <userParam name="5 mm inverse detection cryoprobe"/>
        </instrumentConfiguration>
    </instrumentConfigurationList>
    <acquisition>
        <acquisition1D>
            <acquisitionParameterSet numberOfScans="160" numberOfSteadyStateScans="0">
                <acquisitionParameterFileRefList>
                    <sourceFileRef ref="SOURCE_FILE_1"/>
                    <sourceFileRef ref="SOURCE_FILE_2"/>
                </acquisitionParameterFileRefList>
                <sampleAcquisitionTemperature unitName="kelvin" unitCvRef="UO" value="299.15" unitAccession="UO:0000012"/>
                <spinningRate unitName="hertz" unitCvRef="UO" value="0" unitAccession="UO:0000106"/>
                <relaxationDelay unitName="second" unitCvRef="UO" value="22.2737024" unitAccession="UO:0000010"/>
                <pulseSequence>
                    <cvTerm cvRef="NMRCV" accession="??" name="??"/>
                    <pulseSequenceFileRefList>
                        <pulseSequenceFileRef ref="PULSE_SEQ_SRC"/>
                    </pulseSequenceFileRefList>
                </pulseSequence>
                <DirectDimensionParameterSet numberOfDataPoints="65536" decoupled="true">
                    <acquisitionNucleus cvRef="??" accession="??" name="H1"/>
                    <gammaB1PulseFieldStrength unitName="hertz" unitCvRef="UO" value="344827.586207" unitAccession="UO:0000106"/>
                    <irradiationFrequency unitName="hertz" unitCvRef="UO" value="599.8311617" unitAccession="UO:0000106"/>
                    <decouplingMethod cvRef="NMRCV" accession="NMR:1000046" name="??"/>
                </DirectDimensionParameterSet>
            </acquisitionParameterSet>
            <fidData byteFormat="Complex128" encodedLength="324160"
compressed="true">eJwMl4dfzl8Ux7U3lYZKy0qiomQLPd9zHlmhslMJ2aIoGe29hyhKKURIaWhoPd9zn6zEz46skFKKZMXPX3DPva/PPZ/3u9xlqVhxpdX8Udd0Gp02JIodL2UIN34
ZHCQ7lCseEBM/E981xh2fNnbNDx0yCwLRF+XBYhvrlcna1elI65+TUsveYme2fQhHJzu9m1dwHiesWzwuy5P1n4kE842P2UcKvcFvH6tzw2ph5il74/YB/szWjUrhEkHrma3bO9wDQ7H6
cNmaSzq4aw7rnzxFuFVZT6xbEUxOGjNA/DiVLdq5ndW4DcUpJqmMqVmL85zjhc2GjHnur8DHsmeFG9YfF9errkCm+R/3u3SpOHZBONb+icajsR/Zy5aTrNVohvAtkxbfbH+EXY3uQv9qW
```

Figure 8: The screenshot illustrates how 1D acquisition parameters are stored in the example XML and how raw FID data is stored as a binary blob (base64 encoded binary data, bottom). The sourceFile description (Fig. 7) is here instantiated by a procpar file that is described via CV terms (upper third section) from the nmrCV in cvTermType mode.

Further nmrML data examples

Please keep in mind that the examples on the nmrML.org website are kept updated; so please consider to browse these first, especially if you encounter differences in the following examples to what you see in your updated files and examples.

Example for raw data (MTBLS1)

In the following we explain the nmrML usage in an example that represents a prototypical metabolomics experiment involving the analysis of a complex biofluid. The MTBLS1 experiment (http://www.ebi.ac.uk/metabolights/MTBLS1) investigated urinary metabolic changes in Human type 2 diabetes mellitus patients (21). Figure 9 illustrates how the XSD is instantiated by specifying source files, 1D acquisition[3] and raw FID data in an nmrML xml instance for one of the samples from ADG10003u_007.nmrML in MTBLS1. The example shows how 1D spectral acquisition parameters (as required by the MSI CIMR reporting standard, like probe type, acquisition temperature, water suppression method and pulse sequence) are captured.

```
?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<nmrML xmlns="http://nmrml.org/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0.rc1"
xsi:schemaLocation="http://nmrml.org/schema http://nmrml.org/schema/v1.0.rc1/nmrML.xsd">
 <cvList> <cv id="NMRCV" fullName="Nuclear Magnetic Resonance CV" version="1.0.rc1" URI="http://nmrml.org/cv/v1.0.rc1/nmrCV.owl"/>
 <cv id="UO" fullName="Unit Ontology" version="3.2.0" URI="http://purl.obolibrary.org/obo/"/>
 <cv id="CHEBI" fullName="Chemical Entities of Biological Interest Ontology" version="105" URI="http://purl.obolibrary.org/obo/"/>
 <cv id="NCIThesaurus" fullName="NCI Thesaurus" version="" URI="http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#"/> </cvList>
 <fileDescription>

 <fileContent><cvParam cvRef="NMRCV" accession="NMR:1400165" name="1D NMR acquisition parameter set"/></fileContent>
 </fileDescription>
 <contactList> <contact id="ID00001" fullname="mse" email="mse@UKT01948"/> </contactList>
 <sourceFileList> <sourceFile id="ID00002" name="fid" location="file:/C:/nmrML/examples/MTBLS1/ADG10003u_007/10/fid"
 sha1="4290133c4eadf6f6abcbb236257e240c93df07f9">
 <cvParam cvRef="NMRCV" accession="NMR:1400320" name="Bruker UXNMR/XWIN-NMR format"/>
 <cvParam cvRef="NMRCV" accession="NMR:1400119" name="FID file"/> </sourceFile>
 <sourceFile id="ID00004" name="acqus" location="file:/C:/nmrML/examples/MTBLS1/ADG10003u_007/10/acqus">
 <cvParam cvRef="NMRCV" accession="NMR:1400320" name="Bruker UXNMR/XWIN-NMR format"/>
 <cvParam cvRef="NMRCV" accession="NMR:1002006" name="acquisition parameter file"/> </sourceFile> [OMMISSION HERE]
 </sourceFileList> [OMMISSION HERE]
 <instrumentConfigurationList><instrumentConfiguration id="ID00006">
 <cvParam cvRef="NMRCV" accession="NMR:1400198" name="Bruker NMR instrument"/>
 <userParam name="ProbeHead" value="5 mm PATXI 1H-13C/15N XYZ-GRD Z561501/0002"/>
 <softwareRef ref="ID00005"/> </instrumentConfiguration>
 </instrumentConfigurationList>
 <acquisition> <acquisition1D> <acquisitionParameterSet numberOfSteadyStateScans="8" numberOfScans="128">
 <contactRefList> <contactRef ref="ID00001"/> </contactRefList> <softwareRef ref="ID00005"/>
 <sampleContainer cvRef="NMRCV" accession="NMR:1400132" name="Sample-tube"/>
 <sampleAcquisitionTemperature value="300.000000" unitAccession="UO_0000012" unitName="kelvin" unitCvRef="UO"/>
 <spinningRate value="0" unitAccession="UO_0000169" unitName="dimensionless" unitCvRef="UO"/>
 <relaxationDelay value="3.000000" unitAccession="UO_0000010" unitName="second" unitCvRef="UO"/>
 <pulseSequence> <userParam name="Pulse Program" value="noesypr1d"/> </pulseSequence>
 <shapedPulseFile ref="ID00003"/>
 <DirectDimensionParameterSet decoupled="true" numberOfDataPoints="65536">
 <acquisitionNucleus cvRef="CHEBI" accession="CHEBI_49637" name="hydrogen atom"/>
 <effectiveExcitationField value="699.870000" unitAccession="UO_0000325" unitName="megaHertz" unitCvRef="UO"/>
 <sweepWidth value="14005.602241" unitAccession="UO_0000106" unitName="hertz" unitCvRef="UO"/>
 <pulseWidth value="7.750000" unitAccession="UO_0000029" unitName="microsecond" unitCvRef="UO"/>
 <irradiationFrequency value="699.873291" unitAccession="UO_0000325" unitName="megaHertz" unitCvRef="UO"/>
 <decouplingNucleus cvRef="CHEBI" accession="CHEBI_49637" name="hydrogen atom"/>
 <samplingStrategy cvRef="NMRCV" accession="NMR:1000349" name="uniform sampling"/>
 </DirectDimensionParameterSet></acquisitionParameterSet>
 <fidData compressed="true" encodedLength="188039" byteFormat="Complex128"> [BINARY BLOB OMITTED HERE] </fidData>
    </acquisition1D>
 </acquisition>
</nmrML>
```

Figure 9: Besides illustrating how CVs are referenced in the CVList element at the beginning of a file, the screenshot illustrates how basic acquisition parameters are stored as CV terms or free text user parameters and how raw FID data is stored as base64 encoded binary data. The file size remains small enough to be transferable via web or email. The sourceFile description references a procpar file that is described via CV terms (upper third section) from the nmrCV in cvTermType mode (explained in first row of Tab. 2).

[3] Parameters that are also used in 2D acquisitions can be specified under DirectDimensionParameterSet, e.g. acquisitionNucleus, irradiationFrequency and sampling strategy.

Molecule identification and quantification assignment metadata (spectrum annotation)

NMR vendor formats may store integral values of peaks, but usually do not store the absolute or relative quantity of the identified molecules in the solution. In nmrML, quantification and identification descriptors (Fig. 10) for 1D spectra allow capturing assayed **molecule representations based on CMLSpect** (22) **via atom and bond arrays describing** molecules. These can then be used for annotation and peak assignments in a spectrum, e.g. via nmr-Assign in the Baysil tool suite, and then be stored together with the spectrum in nmrML. This also enables the indication of absolute or relative quantity of a molecule in a sample. These annotated spectra can then be visualized, e.g. using the JavaScript spectral viewer called “JSpectraViewer” (JSV), (Sajed et al. 2016).

```
<spectrumAnnotationList> <quantification spectrumRef="spectrum1">
 <quantificationMethod cvRef="NMRCV" accession="NMR:1000180"
 name="spectral quantitation algorithm"/>
 <quantifiedCompoundList> […]
 <quantifiedCompound name="Hypoxanthine">
 <identifierList> <databaseIdentifier identifier="HMDB00157"
 URI="http://www.hmdb.ca/metabolites/HMDB00157"/> </identifierList>
 <concentration value="0.0228" unitName="micromolar" unitAccession="UO_0000064"unitCvRef="UO"/>
 <peakList> <peak center="0.0" amplitude="0.15439" width="0.003529394412148217"/>
 <peak center="0.0" amplitude="0.14618" width="0.0042547889861630295"/> </peakList>
 </quantifiedCompound> […] <quantifiedCompound name="Methanol">
 <identifierList> <databaseIdentifier identifier="HMDB01875"
 URI="http://www.hmdb.ca/metabolites/HMDB01875"/> </identifierList>
 <concentration value="0.1015" unitName="micromolar" unitAccession="UO_0000064" unitCvRef="UO"/>
 <peakList> <peak center="0.0" amplitude="0.42587" width="0.002386332476445396"/> </peakList>
 </quantifiedCompound>
 </quantifiedCompoundList> </quantification> </spectrumAnnotationList>
```

Figure 10. nrmML xml code mockup example from https://github.com/nmrML/nmrML/tree/master/examples/quantification_example, illustrating the identification and quantification of a subset of two metabolites (Hypoxanthine and Methanol) in a Human serum sample from HMDB.

When is CV-term-referencing advisable?

The XSD branches out into CV-usage, where ...

- The terms are unstable, variant & dynamically evolving, or need to be changed and updated often. E.g. referring to fast paced changing terms such as software names/versions, processing parameters etc. This means that the terms are better maintained by a fast reacting NMR user community rather than by resource-limited Cosmos work packages.
- The terms describe contextual metadata, rather than concrete NMR raw data, i.e. for cases where the terminology is already extensively defined in existing ontologies or CVs.
- The terms represent search attributes for data querying and large scale database-integration in an open linked data fashion.
- The terms should be exploited by profiting from robust subsumption, e.g. exploiting the taxonomic CV backbone to generalize over query attributes and hence increase result recall and precision. Or the terms should be accessible to rule-based or other reasoning techniques and validation.

The CV term referencing mechanism

At some leaf-situated locations, the user is allowed to describe his/her data by means of standardized ontology terms. In the case where the nmrML file is autogenerated from the vendor files, the appropriate CV terms are inserted by the converter, as taken from the parser internal mapping file (See Sec 'How are the Vendor parameters mapped onto nmrML element ?', below). References to all CVs used in an XML instance are recorded in the 'cvList' element at the top of the file, allowing unambiguous non-redundant references to CV term source vocabularies. The XSD branches out into CV-usage, where the terms are unstable, variant & dynamically evolving, or need to be changed and updated often. This applies to fast paced changing terms such as software names/versions, processing parameters etc. In practice, the terms are better maintained by a fast reacting NMR user community rather than by temporary enabled and resource-limited research project affiliates.

We here outline how CV term usage in an nmrML is specified in the XSD. The requirement and modality for a particular CV term occurrence in an XML instance is defined by CV reference types in the XSD, as illustrated in Table 2. Only the UserParamType captures unrestricted free text and makes no CV term reference to populate an XML element.

Table 2: The different ways to reference a CV term from within an nmrML xml file. They differ in the detail they capture, i.e. from single CV Terms up to a complex combinations of CV Terms with parameters and units.

Reference Type	Definition	Attributes	Comment
/TermType	ements of this type hold additional controlled ta or annotation as a simple CV term with no rther values (parameters) associated with it. Only ntrolled CV terms from the cvList specified at the p of the file are allowed here.	/Ref (the ntology), cession (the), name (the bel of the CV rm)	e "CVRef" attribute contains an ID for the urce CV, unique to the XML instance, and hich is defined in the cvList element. This lows for multiple CVs to be referenced crisp d unambiguously. The "accession" tribute contains the ID of the CVterm which unique within the CV. The "name" attribute ntains the term itself, the intelligible label hich is e.g. displayed to a user.
/ParamType	ements of this type hold additional data companying a CV term. In contrast to /TermType, here a pair of descriptors, i.e. the CV rm AND a value (=Parameter) is captured.	/Ref, accession, me, value	e 'value' attribute stores the parameter to captured as value.
/ParamWithUnitType	ements of this type hold additional data or notation, i.e. a controlled term describing a rameter, as well as a value and a description of e unit the value is recorded in. The unit ontology typically used to provide the terms for the unit.	/Ref, accession, me, value, itCVRef, itAccession, itName	e 'unitCvRef', 'unitAccession' and nitName' attributes are used in the same ay to describe the unit as the 'cvRef', ccession' and 'name' terms are used to scribe other CVTerms.
serParamType	e only element allowed to hold uncontrolled er-specified parameters (essentially allowing free xt Strings). For cases where no suitable CV term ists. Before using these, one should verify hether there is an appropriate CV term available, d if so, use the CV term instead. This list can wever later be exploited to generate rresponding term requests in given ontologies or	me, valueType, lue, itAccession, itName, itCvRef	be used sparsely as all descriptors except e unit CV ones, can not easily be exploited computers as their meaning is not rmally accessible.

	/s.		

Besides these primary reference types there are secondary reference types that are compositions of the primary ones, namely ParamGroupType and ReferenceableParamGroupType and lists thereof.

How are FIDs encoded ?

The nmrML converter can correctly convert NMR Bruker/Varian spectra, from their FID to nmrML format. It assumes that the data size must be a power of 2. FID data is stored in nmrML as a binary blob (base64 encoded binary data). Byte ordering is Intel-style little endian. Computers using a different endian style must convert to/from little endian when writing/reading nmrML. The FID should be converted into an array of complex numbers before encoding.

How are nmrML files are being generated ?

Use the nmrML JAVA converter. There is an online version at http://nmrml.org/converter where you upload your bruker or Agilent FID and acqus data and get an nmrML file for the raw data back.

However there is a more comprehensive commandline-based version of the converter, that can convert both Acquition and Processing parameters along with their corresponding spectrum data (See https://github.com/nmrML/nmrML/tree/master/tools/Parser_and_Converters/Java/converter).

How are nmrML files are being populated with values by the vendor to nmrML converter ?

Values and attributes for elements that appear in an nmrML file can be:

- autogenerated at run-time form the source vendor file parameters via simple mappings defined in the parser write files (see supplement).
- autogenerated at run-time by computation from given parameters in the vendor data files. An example is the value for gamma_b1_pulse_field_strength, which is computed from the Varian pulseWidth parameter.
- autogenerated totally anew at parser run time. These refer to mainly administrative information, e.g. local variables like hashes and temporary IDs.

Manually/user generated additional annotations that pertain to metadata that are not easily accessible or not available from the vendor files. This can be done on autogenerated files for semantical enrichment, i.e. quantification, identification and peak assignments.

How are the Vendor parameters mapped onto nmrML element ?

Where can I find a mapping table, where my known vendor file parameters are mapped onto nmrML elements ? You can find the mapping details (acquisition parameter mapping EXCEL file) as a table under

https://docs.google.com/spreadsheets/d/1ZnPmqYZBuI8755UNePRBwVJnLGIQB7D-bO7voA65noU/edit#gid=0

Conversion of Vendor Formats to nmrmL in batch mode

A shell script for cmd based batch processing of multiple Zipped data sets can be accessed at https://github.com/nmrML/nmrML/blob/master/examples/reference_spectra_examples/MMBBI/batch_script.sh

Example files can be found at https://github.com/nmrML/nmrML/tree/master/examples/reference_spectra_examples/MMBBI

Where does the converter know the mappings from vendor parameters to nmrML elements and **CV terms**?

The Vendor2nmrML.java code lies at

https://github.com/nmrML/nmrML/tree/master/tools/Parser_and_Converters/Java/converter/src/org/nmrml/converter

And

https://github.com/nmrML/nmrML/tree/master/tools/Parser_and_Converters/Java/converter/src/org/nmrml/parser

For ontologisation (CV-based standardisation) of vendor parameter values like PPM to UO_0000169:parts per million, the Java parser exploits a parameter to CV term mapping file at

https://github.com/nmrML/nmrML/tree/master/tools/Parser_and_Converters/Java/converter/src/resources

e.g.

https://github.com/nmrML/nmrML/blob/master/tools/Parser_and_Converters/Java/converter/src/resources/onto.ini

specifies:

```
#units

PPM = UO_0000169;parts per million

HERTZ = UO_0000106;hertz

SECOND = UO_0000010;second

...

[NMRCV]

#Instrument

BRUKER = NMR:1400198;Bruker NMR instrument

VARIAN = NMR:1400234;Varian NMR instrument

# Sofware data format

BRUKERFORMAT = NMR:1400320;Bruker UXNMR/XWIN-NMR format

VARIANFORMAT = NMR:1400297;Varian VNMR format
```

```
# Generic Sofwares

PREPROC = NMR:1400134;spectrum pre-processing software

POSTPROC = NMR:1400135;spectrum post-processing software

# Commercial Sofwares

UXNMR = NMR:1400320;Bruker UXNMR/XWIN-NMR format

XWIN-NMR = NMR:1400320;Bruker UXNMR/XWIN-NMR format

TOPSPIN = NMR:1400215;Bruker TopSpin software

VNMRJ = NMR:1000277;VnmrJ software

# part of NMR Instrument

NMR_PROBE = NMR:1400014;NMR Probe

TUBE = NMR:1400132;Sample-tube

#NMR sampling strategy

UNIFORM_SAMPLING = NMR:1000349;uniform sampling

NON-UNIFORM_SAMPLING = NMR:1000350;non-uniform sampling

SPINNING_RATE = NMR:1400028, spinning rate

# Decoupling

OFF_DECOUPLE = NMR:1000055;off resonance decoupling
```

For the TMIC Python parser this mapping is specified in the writer file at:

https://github.com/nmrML/nmrML/blob/master/tools/Parser_and_Converters/python/pynmrml/io/writers/nmrml_writer.py

E.g. for the acquisition parameter ontologisation, we find:

```
  acquisition_1D = Acquisition1DType()
    param_set = AcquisitionParameterSet1DType(
        numberOfScans = self.reader.number_of_scans(),
        numberOfSteadyStateScans = self.reader.number_of_steady_state_scans() )
    param_set.set_sampleAcquisitionTemperature( ValueWithUnitType(
      value = self.reader.sample_acquisition_temperature(),
      unitName = "kelvin", unitAccession = "UO:0000012", unitCvRef = "UO" ))
    param_set.set_spinningRate( ValueWithUnitType(
        value= self.reader.spinning_rate(),
        unitName="hertz", unitAccession="UO:0000106", unitCvRef="UO" ))
    param_set.set_relaxationDelay( ValueWithUnitType(
      value= self.reader.relaxation_delay(),

      unitName="second", unitCvRef="UO", unitAccession="UO:0000010" ))
```

How does the transformation and **recomputation of vendor formatted data values** into nmrML work ?

Here an example how a Varian Degree Centigrade value is converted to Kelvin in the Python parser at https://github.com/nmrML/nmrML/blob/master/tools/Parser_and_Converters/python/pynmrml/io/readers/varian_reader.py

```
class VarianOneDReader(AbstractReader):

    def load_params(self):

        dic, _ = nmrglue.varian.read(self.input_dir)

        self.varian_params = dic["procpar"]

  […]

    def sample_acquisition_temperature(self):

        # need to output in kelvins, but the varian format stores it

        # in C so we need to convert

        return convert.celcius_to_kelvin(self.get_param('temp'))

        #return format( float(self.get_param('temp')) + 274.15 )
```

Selecting good example NMR data sets for nmrML xml instances

We defined characteristics of intelligible/intuitive example data set:

- The data was gathered in a prototypical, abundant experiment set up, representative for metabolomics data acquisition
- The data should stem from a simple experimental set-up (e.g. 1D 1H NMR data)
- The data has a published paper available (not a method-, but a research-paper)
- The data has a database entry available, e.g. in MetaboLights or HMDB
- The data has accompanying original data files (FIDs)
- The data is using an abundant vendor format like Bruker or Agilent/Varian standard files
- The data is associated with a responsive contact person, in case someone needs to get back to the data producers to be able to gather additional information or resolve questions
- The data has been analyzed further with open source tools like Batman or MetaboQuant, so that we can later reproduce the same results based on the converted nmrML data.

Bruker and Varian NMR raw data nmrML examples

According criteria outlined in D2.4 Appendix D in (http://www.cosmos-fp7.eu/system/files/presentation/COSMOS%20DELIVERABLE%202.4_0.pdf) we have collated example NMR raw data sets to be converted into nmrML by an automatic parser that reads in the vendor files and writes valid nmr xml files which comply with the nmrML.XSD. These full examples nmrML xml files can be found under http://nmrml.org/examples. These example instances can be found in the corresponding github ‘example’ folder, together with an accompanying readme file illustrating its generation or on the documentation page at nmrml.org/schema.

NMR Spectra Viewer examples

The T Ebbels Group (Imperial College) developed a Matlab based viewer:

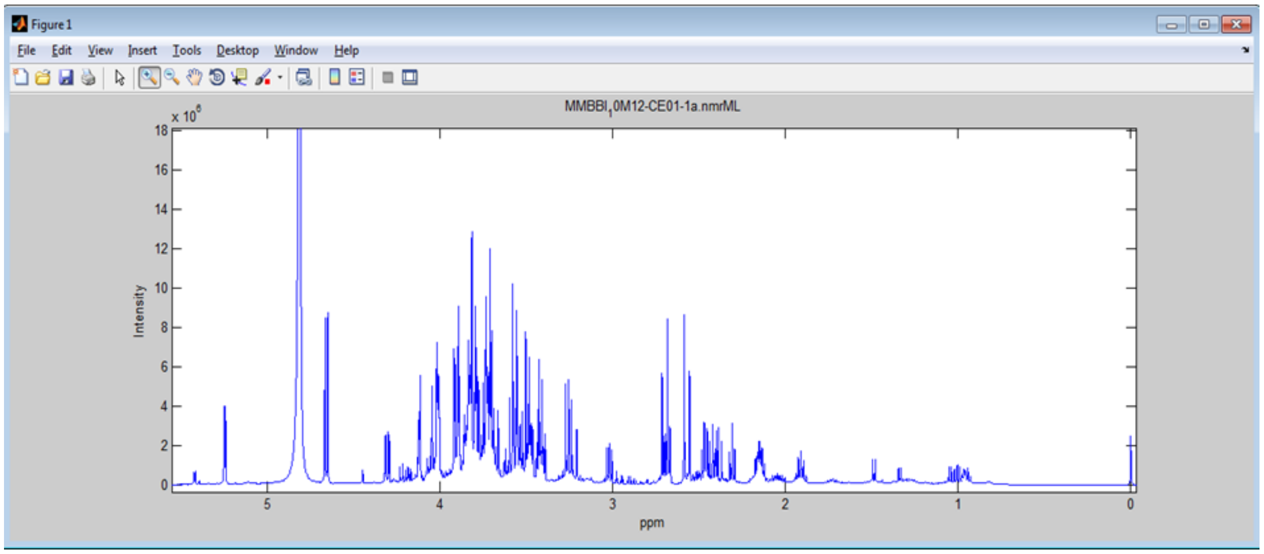


Figure 11: A spectrum plot generated via the Matlab nmrML parser from a nmrML file (MMBBI_10M12-CE01-1a.nmrML issued from DOI:10.1093/jxb/ert349). It reads all fields of nmrML parameters, and decodes FID and spectrum array data.

Some resolved XSD design Issues to note

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 here the issues and why the decision that is implemented was made.

CV exchange syntax

We choose the OBO format[4] as exchange syntax for the CV, as the OBO tools are less stable. Because the OBO format is only established in the biology domain (lack of off-the-shelf development tools) and there are hence less resources to integrate with, in particular with OWLs predominance for the semantic web.

Count attributes

At the moment all list elements would have a count attribute. The reason is that parsers implemented in languages where memory allocation or array sizing is important, it is a nice performance enhancement to have a count attribute indicating how many elements there are in the list. As this is an easy target for creating inconsistent files (i.e. specifying a count=”5” attribute followed by 6 items in the list), this was deemed to be not very useful. That is where we diverge from our role model the PSI mzML not using count attributes in the next versions.

Numerical value and datetimestamp encoding

[4] http://www.geneontology.org/GO.format.obo-1_2.shtml

All numerical values shall appear in the XML schema datatype specification (http://www.w3.org/TR/xmlschema-2/). The number 1/10 must always appear as 0.10 and never as 0,10. A + before a number (+5.0) is prohibited and left implicit.

Datetimestamps must also be encoded as in the XML specification such as 2007-06-27T15:23:45.00035.

All ID attributes follow the XML schema datatype xs:ID (http://www.w3.org/TR/xmlschema-2/#ID), which means that no two id attributes may be the same within a document, and id attributes must be purely alphanumerical strings with at least one letter. Thus, they may not contain spaces or underscores, and id attributes may not be a plain number.

Pulse Sequences

An area difficult to capture were the pulse sequences, which can be completely customized by a user. In this case we opted to capture the names of several of the most common pulse sequences in the CV or alternatively allowing a reference (via a URI) to the pulse sequence program source code. While not readable in a vendor agnostic manner this decision still allows for most experiments to be reproduced, while also allowing more custom information to be captured.

Related specifications

Related specifications include the following:

mzML: http://www.psidev.info/mzml_1_0_0. The XML based open and PSI approved Mass Spectrometry data standard. This format served as a general role model for the development of nmrML and its success made us to copycat many of their design decisions.

ISA-Tab: http://isa-tools.org. As research in biomedical and life sciences is increasingly moving towards multi-omics studies, the 'Investigation/Study/Assay' ISA-Tab format was developed to represent experimental metadata independently from the assay technology used. We will use ISA-Tab to standardize metabolomics reporting requirements and terminologies through customized ISA configurations. The 'Investigation/Study/Assay' ISA-Tab format was developed to represent experimental metadata independently from the assay technology used. We will use ISA-Tab to standardize metabolomics reporting requirements and terminologies through customized configurations. We will build on the BioSharing and the ISA-Tab efforts to harmonize representation of the metadata recommendations with other -omics communities, and use automated tests to ensure the interoperability of the metadata between the involved data producers, -consumers and -repositories. The EBI, IPB and MRC will be working with the UOXF to create both core and extended configurations (specific to the research discipline and technologies) suitable for metabolomics, in compliance with the annotation manual created in WP4.

Analytical Information Markup Language (AnIML): http://animl.sourceforge.net . An emerging ASTM XML standard for analytical chemistry data. AniML captures the instrumental set-up of chemical analytical techniques, so is more general in scope than our NMR restricted effort.

Pipelines and Workflow support

A Galaxy-based (https://galaxyproject.org/) NMR data analysis pipeline (Fig. 12) is developed at University of Liverpool that uses nmrML as its main raw spectral data format. The tameNMR pipeline consists of modules for NMR data processing, including normalisation, referencing, alignment and binning, as well as the most popular univariate and multivariate statistical analyses used in metabolomics. It also contains visualisation tools aimed at the identification of significant regions in the spectra. tameNMR already covers the backbone of a standard NMR workflow, with nmrML being used for all raw, processed and annotated NMR data storage throughout the pipeline. Further, the modules can be used as standalone tools or re-combined if needed, as the nmrML converters and readers are already available as standalone modules. The outputs of the pipeline, or any module, can easily be downloaded and shared between research groups or through public repositories. Each module is designed to take nmrML as input and, where appropriate, produce nmrML metadata outputs. The outputs of the pipeline, or any module, can easily be downloaded and shared between research groups or through public repositories. The pipeline is still under development and early stage code can be obtained from http://github.com/pgb-liv/tameNMR.

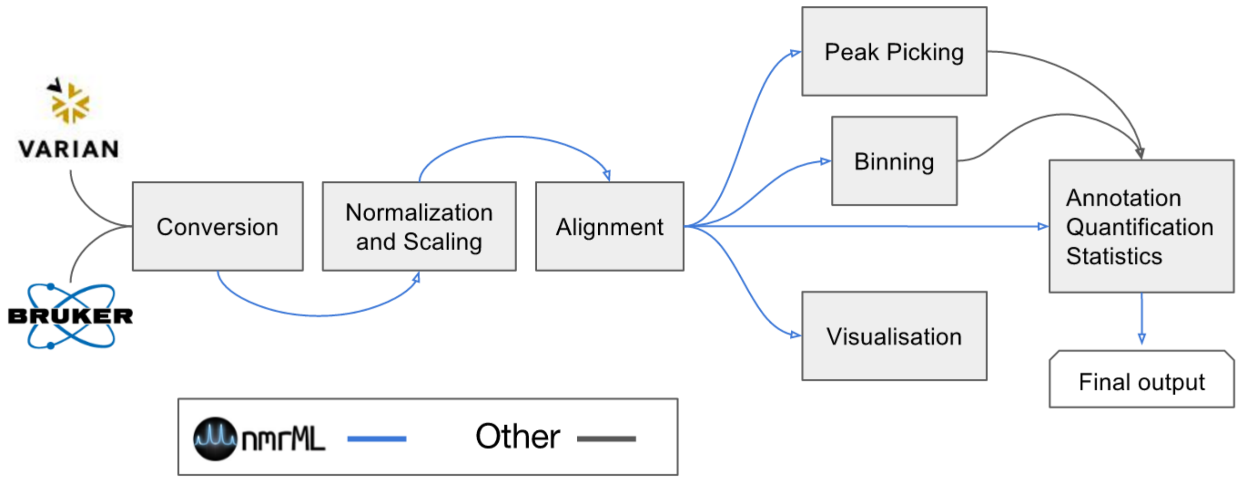


Figure 12. NMR data analysis workflow as a Galaxy tool pipeline. Each grey rectangle represents a functional module (or a group of modules) in the pipeline. The NMR data flows between modules in nmrML format (blue lines) allowing easy addition and substitution of modules.

Relation to other NMR formats

NMR-STAR is the data format used by BMRB [*E. L. Ulrich, D. Argentar, A. Klimowicz, and J. L. Markley, "STAR/CIF macromolecular NMR data dictionaries and data file formats," Acta Crystallographica Section A Foundations of Crystallography 52(a1):C577-C577 (1996)*], and it is fully compatible with the mmCIF format used in the Protein Data Bank (PDB) archive. NMR-STAR and mmCIF deal with data derived from NMR and associated metadata rather than the raw data itself.

Efforts are underway to create **Linked Open Data (LOD) RDF versions** of data standards to facilitate cross database querying via federated SPARQL queries, e.g. in the future a BMRB endpoint (26) might be queried together with a MassBank (27) endpoint[5].

[5] https://github.com/sneumann/SemanticMetabolomics/wiki/01-Home