MSA / MAS / AMAS
Hyper-Dimensional Data File Specification

Version 1.0
February 2016

Prologue

The MSA / MAS / AMAS HyperDimensional Data File (HMSA, for short) is intended to be a common format to permit the exchange of hyper-dimensional microscopy and microanalytical data between different software applications. The expected applications include:

- Hyper-spectral maps, such as electron energy loss spectroscopy (EELS), energy-dispersive x-ray spectrometry (XEDS), or cathodoluminescence spectroscopy (CL).
- 'Hyper-image' maps, such as pattern maps using electron backscatter diffraction (EBSD) or convergent beam electron diffraction (CBED).
- 3-dimensional maps, such as confocal microscopy, or focussed ion beam (FIB) serial section maps.
- 4-dimensional maps, such as double-tilt electron tomography.
- Time-resolved microscopy and spectroscopy.

In addition to storing hyperdimensional data, the HMSA file format is suitable for storing conventional microscopy and microanalysis data, such as spectra, line profiles, images, and quantitative analyses, as well as experimental conditions and other metadata.

I. Current status

This document defines Version 1.0 of the MSA/MAS/AMAS HyperDimensional Data File format, and supersedes all prior draft specifications.

II. Contributors

The MSA/MAS/AMAS HyperDimensional Data File format specification was developed by the HMSA Working Group of the Standards Committee of the Microscopy Society of America (MSA), including members of the MSA, the Microanalysis Society (MAS), and the Australian Microbeam Analysis Society (AMAS). The specification presented in this document is based on contributions from:

- Nestor J. Zaluzec, Committee Chair (Argonne National Laboratory)
- Mike Kundmann (e-Metrikos)
- Aaron Torpy (CSIRO Australia)
- Nicholas C. Wilson (CSIRO Australia)
- Colin M. MacRae (CSIRO Australia)

III. Additions

It is expected that additional definitions for experimental conditions or datasets will be required to facilitate the broader adoption of the HMSA format, particularly with experimental techniques not well covered by the initial range of datasets and conditions defined in Appendix A and Appendix B. To propose additional templates or classes for datasets or conditions, please follow the HMSA submission procedure on the MSA website at:

http://wwww.microscopy.org/HMSA/

Submissions will be periodically reviewed by the HMSA working group, and accepted or amended HMSA dataset and condition templates will be published at the above URL.

Contents

- Prologue
  - I. Current status
  - II. Contributors
  - III. Additions
- 1. Overview
  - 1.1 Design considerations
  - 1.2 Binary and XML file pair
    - 1.2.1 HMSA general structure
    - 1.2.2 XML general structure
    - 1.2.3 HMSA-XML association
  - 1.3 HyperDimensional data
  - 1.4 Unicode and internationalisation
  - 1.5 Minimalism
  - 1.6 Extensibility
  - 1.7 What HMSA does not do
- 2. XML file specification
  - 2.1 XML file overview
  - 2.2 XML specification
    - 2.2.1 XML features not supported
    - 2.2.2 XML conformance and validation
    - 2.2.3 Character encodings
    - 2.2.4 Byte order markers
    - 2.2.5 Case sensitivity
  - 2.3 XML declaration
    - 2.3.1 The XML `version` attribute
    - 2.3.2 The XML character `encoding` attribute
    - 2.3.3 The XML `standalone` attribute
  - 2.4 Document root element
    - 2.4.1 The `Version` attribute
    - 2.4.2 The `xml:lang` attribute
    - 2.4.3 The `UID` attribute
  - 2.5 XML element formatting
    - 2.5.1 Numerical data types
    - 2.5.2 Arrays of values
    - 2.5.3 Numerical values

        - 2.5.4 Physical units
        - 2.5.5 Alternative language attributes
        - 2.5.6 Special characters
        - 2.5.7 Ordering of elements
- 3. The `<Header>` list element
    - 3.1 Header items are optional
    - 3.2 The `<Checksum>` element
    - 3.3 The `<Title>`, `<Author>` and `<Owner>` elements
    - 3.4 The `<Date>`, `<Time>` and `<Timezone>` elements
    - 3.5 The `<ArbitraryData>` element
    - 3.6 Other optional header elements
- 4. The `<Conditions>` list element
    - 4.1 Conditions are optional
    - 4.2 Condition templates and classes
    - 4.3 Condition identifiers
- 5. The `<Data>` list element
    - 5.1 Dataset templates and classes
    - 5.2 The `<DataOffset>` and `<DataLength>`elements
    - 5.3 The `<DatumType>` element
    - 5.4 The `<DatumDimensions>` element
        - 5.4.1 The `<Dimension>` element
        - 5.4.2 Datum as single values
        - 5.4.3 Datum as arrays
        - 5.4.4 Datum as 2D arrays
        - 5.4.5 Datum as 3D arrays and higher dimensionality
    - 5.5 The `<CollectionDimensions>` element
    - 5.6 The `<IncludeConditions>` element
- 6. Format of datasets in the HMSA binary file
    - 6.1 Datum-first order
    - 6.2 Order of collection dimensions
    - 6.3 Higher order collection dimensions
    - 6.4 Order of datum dimensions
    - 6.5 Hyperspectral map example
    - 6.6 Coordinate mapping equations
- Appendix A - Dataset templates and classes
    - `<Single>`
    - `<IrregularArray>`
    - `<RegularArray>`

- Appendix B - Condition templates and classes
    - <Acquisition>
    - <Acquisition Class="Single">
    - <Acquisition Class="IrregularArray">
    - <Acquisition Class="RegularArray">
    - <Acquisition Class="RegularArray/Linescan">
    - <Acquisition Class="RegularArray/XY">
    - <Acquisition Class="RegularArray/XY/Z">
    - <Calibration>
    - <Calibration Class="Constant">
    - <Calibration Class="Linear">
    - <Calibration Class="Polynomial">
    - <Calibration Class="Explicit">
    - <Composition>
    - <Composition Class="Elemental">
    - <DateTime>
    - <Detector>
    - <Detector Class="Camera">
    - <Detector Class="Spectrometer">
    - <Detector Class="Spectrometer/CL">
    - <Detector Class="Spectrometer/WDS">
    - <Detector Class="Spectrometer/XEDS">
    - <ElementalID>
    - <ElementalID Class="X-ray">
    - <Instrument>
    - <Material>
    - <Probe>
    - <Probe Class="EM">
    - <Probe Class="EM/TEM">
    - <RegionOfInterest>
    - <Specimen>
    - <SpecimenPosition>
- Appendix C - Units and prefixes
- Appendix D - Unicode character substitutions
- Appendix E - Example HMSA XML files

1. Overview

1.1 Design considerations

The following requirements were considered in the design of this file format:

1. Modern experimental apparatus produce data with high dimensionality, such as a spectral maps, and 3D serial section maps. Therefore, this file format must store data of high dimensionality.
2. High dimensionality data is necessarily very large, and consequently difficult and time consuming to store or transfer over networks. The file format must therefore be as compact as is reasonably practical.
3. Many microanalytical techniques produce structurally similar hyperdimensional data. To simplify implementation of common tools, this file format must use a common format to store data produced by different analytical techniques.
4. The data format must preserve the scientific accuracy and meaning of the data. Therefore, the file format must store data without loss of precision, and include sufficient experimental parameters to permit the correct interpretation of the data.
5. To achieve the intended mission of being a widely-supported exchange format, the file format must achieve acceptance from instrument and software vendors, and from the microanalysis community. Consequently, the file format must be useful, easy to understand, and easy to implement.
6. Furthermore, as the file format is intended for exchange, it must be readable (and implementable) in any commonly available programming languages and environments. The format must therefore be platform independent, and not require any proprietary or special software or hardware.

1.2 Binary and XML file pair

To satisfy the above requirements, the MSA/MAS/AMAS Hyperdimensional Data File format uses a pair of files; a simple binary file to efficiently store the experimental data, and a text-based XML file to store the experimental conditions. The advantages of this dual format are:

- The structure of the binary file format is simple, unambiguous, and precisely defined in a human readable format within the XML file.
- High dimensionality experimental data is binary encoded for space efficiency, whilst also being easy to read and write programmatically.
- Experimental conditions are stored in a human-readable and self-descriptive format. Conditions are stored in a hierarchical structure to logically classify related settings.
- No special libraries are required to read or write HMSA/XML files. For convenience, XML libraries may be used, and are freely available on most programming environments.

1.2.1 HMSA general structure

The HMSA file is a binary file format consisting of an 8 byte (64 bit) unique identifier (See Section 2.4.3: The `UID` attribute), followed by one or more dataset objects. The location, size and layout of the binary dataset objects are described in the dataset definitions within the XML file (See Section 5: The `<Data>` list element, and specifically Section 5.7: Format of dataset in HMSA binary file), and are not described within the binary HMSA file. The values contained within the HMSA file datasets cannot therefore be read or interpreted without the corresponding dataset definition within the XML file.

Blocks of arbitrary and proprietary binary or text data also may be placed in the binary HMSA file. These arbitrary data blocks may be used to store proprietary application-specific data, or ancillary experimental data that cannot be formatted as a HMSA data set object (See 5. The `<Data>` list element). The formatting of these arbitrary data blocks in the HMSA file are not defined by this specification, but the location and size of the arbitrary block should be declared in the `<Header>` section of the XML file using one or more `<ArbitraryData>` elements (See Section 3.5: The `<ArbitraryData>` element).

The byte ordering of the HMSA binary file *shall* be little-endian (Intel/Windows style).

1.2.2 XML general structure

The XML file consists of human-readable hierarchical text, using a subset of the XML version 1.0 format.

The structures within the XML file are strictly defined and self-descriptive, so that the XML file can be read and interpreted correctly without a finely detailed study of the specification. This strict definition does, however, require software that writes the XML files to diligently adhere to the specification.

The structure of the XML file is described in detail in Section 2: XML file specification.

1.2.3 HMSA-XML association

Because the XML file is required to interpret the HMSA file, the HMSA/XML files must be associated in such a way that software that loads a HMSA file can readily and unambiguously locate the associated XML file. The principal method by which the HMSA and XML files are associated together is by file name. The HMSA/XML file pairs *shall* share the same file name except for their file extensions, such as "Spodumene.HMSA" and "Spodumene.XML". The HMSA/XML file pairs *should* be transferred together, and stored in the same directory.

Users may inadvertently rename or move one member of the file pair, which would prevent software from finding the correct experimental conditions or binary data. To reduce this risk, the XML and HMSA files each contain an identifier that is, for all intents and purposes, unique to each individual pair of files. By comparing the unique identifiers (UIDs) given in the XML and HMSA file, software can be assured that binary data matches the description in the XML file, and vice versa. Furthermore, by searching the file system for XML or HMSA files containing the UID, software may automatically find renamed or relocated files. This pseudo-unique identifier is a 64-bit code, providing a possible 2^{64} (~1.84×10^{19}) unique values. The UID is described further in Section 2.4.3: The `UID` attribute.

1.3 HyperDimensional data

The HMSA file distinguishes between two forms of dataset dimensionality:

- Collection dimensionality refers to the spatial or temporal partitioning over which the specimen was analysed, such as a single point analysis (0D), a line scan (1D), an image or XY rastered map (2D), a serial section map (3D), etc.
- Datum dimensionality refers to the dimensions of a single measurement, such as a single-valued pixel in an elemental x-ray map (0D), a spectrum in a hyperspectral map (1D), a diffraction pattern image in a hyperimage map (2D), etc.

The HMSA format supports any combination of collection and datum dimensionality. However, this specification does not require software to implement support for all combinations of collection and datum dimensions. The principle combinations of collection and datum dimensionality envisaged for this file format are summarised in the table below:

	0D datum	**1D datum**	**2D datum**
0D collection	N/A *	A single spectrum acquisition (e.g. EELS point analysis.)	A single 2D image acquisition (e.g. diffraction pattern image) **
1D collection	A linescan or time sequence of single-valued data (e.g. Ti Kα counts, BSE yield, vacuum pressure.)	A linescan or time sequence of spectra.	A linescan or time sequence of 2D data.
2D collection	An X/Y map of single-valued data (e.g. a CCD micrograph)**	An X/Y hyperspectral map (i.e. one spectrum per pixel)	An X/Y ‘hyperimage’ map (i.e. one image per pixel)
3D collection	An X/Y/Z serial section map of single valued data.	An X/Y/Z hyperspectral serial section map	An X/Y/Z hyperimage serial section map.

* Data with 0 collection dimensions and 0 datum dimensions implies a dataset comprised of one single-valued measurement. Single-valued data *should* be stored in the XML file in preference to the HMSA file to maximise readability.

** There is potential for ambiguity when storing a 2D image such as an optical image, BSE image or an EBSD pattern as to whether there should be 2 collection dimensions and 0 datum dimensions, or vice versa. The following principles *should* be followed:

- If the image relates to measurements of the specimen over multiple points in space or time, such as the distribution of an element over a surface, this is a 2D collection of a 0D datum. Use the `<RegularArray>` dataset template with two collection dimensions.
- If the image relates to a single measurement of the specimen at one point in space or time, with a 2D dispersion over the detector such as a diffraction pattern, this is a 0D collection of a 2D datum. Use the `<Single>` dataset template with two datum dimensions.

Further dataset templates are defined in Appendix A.

1.4 Unicode and internationalisation

The HMSA XML file format *requires* the use of the UTF-8 Unicode character encoding, permitting native-language representations of the non-English names for authors, organisations, specimens, locations, etc. However, for maximum interoperability, the names of XML elements and attributes shall be given in US English using the ASCII character set. Furthermore, the values of elements shall be given in US English where possible, with non-English text provided as an alternative translation to the English text using an `alt-lang-[xx][-YY]` attribute (see Section 2.5.5: Alternative language attributes.)

In addition to supporting non-English scripts, the use of Unicode for the HMSA XML file allows the use of scientifically meaningful non-Latin characters such as α, μ, and Å. However, these characters may not be typeable on many standard keyboards, and so they *should only* be used when no unambiguous Latin character equivalent is available. Please refer to Appendix C for a list of permitted Unicode characters in units and unit prefixes.

In cases where the Unicode character set includes multiple code points for visually indistinguishable glyphs, HMSA XML files *shall* consistently use one code point in preference to any alternatives (see Appendix D).

1.5 Minimalism

The purpose of the HMSA file format is to enable the convenient exchange of scientific data between different software packages. To succeed in this purpose, the HMSA file format must be unambiguous in its specification, and easy to implement. To this end, the HMSA XML file format has been designed with a minimalist core of mandatory features that are necessary only to properly determine the *layout* of the hyperdimensional dataset(s) in the HMSA binary data file. The structure of the dataset definition in the XML file is strictly defined, with neither descriptive nor optional features (see Section 5: The `<Data>` list element).

All useful experimental conditions (such as spectrometer gain and offset) and other metadata (such as author or date) are recommended, but optional. Nevertheless, to ensure compatibility, the structure and format of these optional conditions and metadata elements are defined in this document (see Section 3: The `<Header>` list element and Section 4: The `<Conditions>` list element).

The absolute minimum effort possible to produce a conformant HMSA XML file is demonstrated in the '*baseline*' HMSA XML example file in Appendix E. This file contains no optional elements such as conditions or metadata. Important conditions such as microscope settings and spectrometer calibration are not included, meaning that the spectrum can only be interpreted as raw channels, and the user is responsible for determining energy calibration and accelerating voltage. For reference, the same file is also provided in the '*typical*' profile (*ibid*), which includes all common experimental conditions and metadata.

1.6 Extensibility

In addition to being simple and easy to implement (See Section 1.5: Minimalism), a key feature of the HMSA file format is that it is *extensible*. Although this specification enumerates a number of common condition objects (See Appendix B), the specification permits the unlimited use of additional, un-specified experimental conditions to be stored in the HMSA XML file (See Section 4: The `<Conditions>` list element). Critically, the well-formed, hierarchical and self-descriptive nature of XML allows these additional conditions to be included without imposing an additional burden on applications to support any or all of these conditions. In effect, applications are not required to read, write or interpret *any* conditions, but may elect to provide additional scientific meaning or interpretation to the data by including additional conditions to any degree of detail.

For example, consider the case of a typical XEDS spectral map collected in an SEM. A 'typical' HMSA file would include conditions for spectrometer calibration and beam accelerating voltage. This information is sufficient for a basic interpretation of the map data, such as peak identification in spectra and generating elemental region of interest (ROI) images. A more detailed file may also include a Faraday cup beam current measurement, and even intensity measurements from standard reference materials so as to allow quantification of elemental compositions. An extreme example may also include all electron gun conditions, lens currents, and the like, so as to allow the comparison or monitoring of microscope and detector performance between instruments or over time. However, not all SEMs have Faraday cups, and nor do all experiments require quantification or performance monitoring, and thus these elements are purely optional.

In addition to supporting unlimited experimental conditions, the HMSA specification also supports the inclusion of multiple binary datasets in a single HMSA/XML file pair. Typical usage cases for multiple dataset files are:

- The storage of multi-detector maps, such as simultaneous XEDS+EELS in a TEM, XEDS+EBSD in a SEM, or WDS+XEDS+CL in an EPMA.
- The storage of auxiliary map data that is helpful for the interpretation of the primary dataset, such as a beam current/flux map, a specimen thickness map, or a detector saturation/dead-time map.
- The storage of reference spectra with spectral maps.

Support for multiple datasets is provided in such a way as to impose no additional burden on applications that expect only single-dataset files. Applications are not required to support multiple datasets.

1.7 What HMSA does not do

To reduce the complexity of implementing HMSA support, certain features or usage cases have been excluded:

- HMSA is not intended to be a general long-term archival format for all relevant or extraneous data from a set of experiments. HMSA is intended to store the data, and *optionally* the relevant conditions, from a single experiment, on a single apparatus, from a single specimen, collected over a single contiguous time interval.
- No compression is to be used on either the XML or HMSA file, as compression algorithms may be proprietary or unavailable in some environments. Users *may* elect to compress the XML/HMSA file pair for transmission or storage at their own discretion, but HMSA-compatible software *should not* write compressed HMSA/XML files.
- The format is not primarily intended to be an efficient 'working' format for applications, and so it has not been specifically optimised for minimum memory footprint, maximum read/write speed, efficient random seeking, etc.
- HMSA is not intended to support all esoteric or uncommon experimental techniques. Whilst a reasonable effort has been made to support a broad range of experimental dataset types, the HMSA format may not be particularly amenable to some types of experimental data (sparse spectra, for example.)

2. XML file specification

2.1 XML general structure

The XML file consists of human-readable hierarchical text, using a subset of the XML version 1.0 format (see Section 2.2: XML specification). The structures within the XML file are strictly defined and self-descriptive, so that the XML file can be read and interpreted correctly without a finely detailed study of the specification. This strict definition does, however, require software that writes the XML files to diligently adhere to the specification.

The XML files have the following general structure:

- An XML declaration
- An MSAHyperDimensionalDataFile root element, containing:
    - A Header element, containing:
        - Descriptive metadata such as the document title, collection date, author, etc.
    - A Conditions element, containing:
        - One or more items of experimental conditions that describe how the dataset is to be interpreted or displayed, such as microscope and spectrometer settings.
    - A Data element, containing:
        - One or more dataset items, which formally define the address, ordering, and size of the binary data block within the HMSA file.

In XML, this looks like:

```
<?xml version="1.0" [...] ?>
<MSAHyperDimensionalDataFile [...] >
   <Header>
      [...]
   </Header>
   <Conditions>
      [...]
   </Conditions>
   <Data>
      [...]
   </Data>
</MSAHyperDimensionalDataFile>
```

The XML declaration, `<MSAHyperDimensionalDataFile>` document root element, `<Header>`, `<Conditions>` and `<Data>` elements are described in the following sections:

- Section 2.3: XML declaration
- Section 2.4: Document root element
- Section 3: The `<Header>` list element
- Section 4: The `<Conditions>` list element
- Section 5: The `<Data>` list element

2.2 XML Specification

The HMSA XML file specification follows the W3C Extensible Markup Language (XML) 1.0 Recommendation (Fifth Edition), except where noted below (See http://www.w3.org/TR/xml/).

2.2.1 XML features not supported

To simplify the tasks of reading, writing and interpreting HMSA XML files, this specification excludes certain XML features that may complicate implementation for no benefit in this application. HMSA XML files *shall not* contain the following XML feature declared in the XML 1.0 recommendation (section numbers in parentheses):

- Comments (2.5)
- Processing instructions (2.6)
- CDATA sections (2.7)
- Document type definitions (2.8)
- Element type definitions (3.2)
- Conditional sections (3.4)
- Entity declarations (4.2)
- Notation declarations (4.7)

The HMSA XML format also explicitly does not support the following associated W3C XML specifications:

- XML Schema
- Namespaces in XML

2.2.2 XML conformance and validation

The W3C XML specification defines two levels of compliance; *conformant*, and *valid*. Conformant XML files satisfy all requirements of the XML specification, such as well-formedness. Valid XML files are conformant XML files, and also contain document type definitions (DTDs) that specify the structure and range of all elements in the XML file. Valid XML files can therefore be validated for completeness and correctness by a generic validating XML parser, without reference to an external specification of the file format. In effect, valid XML files are self-specifying.

In the interests of minimising the size and complexity of HMSA XML files, XML document and element type definitions were excluded from the HMSA XML specification (See Section 2.2.1: XML features not supported). Consequently, HMSA XML documents are *conformant* XML files, but not *valid* XML files.

2.2.3 Character encodings

HMSA XML files *shall* only be encoded in the Unicode UTF-8 character encoding. To provide backwards compatibility with the ASCII character set, HMSA XML files *should* use the basic Latin characters and symbols in the range of U+0032 to U+007E in preference to visually similar Unicode characters when it is customary to do so, and whenever such substitution does not change the meaning or introduce ambiguity. For example, ‘Ka’ *should* be used to represent the Kα x-ray in the Siegbahn notation, and ‘um’ *should* be use to represent μm. Further character substitutions are specified in Appendix D.

2.2.4 Byte order markers

Byte order markers (BOM) are not required for UTF-8 encoded text files, but may be automatically inserted at the start of the file stream by certain text editors. Thus, HMSA XML files may, but *should not*, contain the UTF-8 BOM (0xEFBBBF), and *shall not* contain byte order markers for other character encodings (e.g. 0xFFFE for UTF-16LE on Windows, or 0xFEFF for UTF-16BE on Unix/Linux/Mac). HMSA XML parsers *shall* process and ignore UTF-8 BOM, if present.

2.2.5 Case sensitivity

As defined in the XML standard, the structure of an XML file is case sensitive. The names of all elements and attributes *shall* be written with the case specified in this document. The values of attributes and elements are also assumed to be case sensitive, unless specified otherwise in this document.

To avoid confusion, identifier attributes such as `Name` and `ID` *shall* have unique values in case-insensitive comparison.

2.3 XML declaration

The HMSA XML file *shall* begin with an XML declaration of the form:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

The attributes of the XML declaration are described below.

2.3.1 XML `version` attribute

The `version` attribute of the XML declaration *shall* have the value "1.0". XML version 1.1 or subsequent versions are not supported by this version of the HMSA/XML specification.

2.3.2 XML character `encoding` attribute

The `encoding` attribute of the XML declaration *shall* have the value "UTF-8". No other character encoding is permitted for HMSA XML files.

2.3.3 XML `standalone` attribute

The `standalone` attribute of the XML declaration *shall* have the value "yes". HMSA XML files do not support external document type definitions.

2.4 Document root element

The root element of the HMSA XML file *shall* be named `<MSAHyperDimensionalDataFile>` and be declared in the following form:

```
<MSAHyperDimensionalDataFile Version="1.0" xml:lang="en-US"
UID="193581B9DD220ABB">
```

The attributes of the root element are described below.

2.4.1 The `Version` attribute

The HMSA version *shall* be declared as "1.0" in the `Version` attribute.

2.4.2 The `xml:lang` attribute

The default language of the document *shall* be US English, which *shall* be declared using an `xml:lang` attribute of the document root element with a value of "en-US".

2.4.3 The `UID` attribute

A pseudo-unique identifier *shall* be provided in the `UID` attribute in the form of 16 hexadecimal characters (0-9, A-F), representing a 64-bit binary value.

The 64-bit unique identifier, which is stored in both the XML and binary HMSA files, serves two purposes:

1. To verify that a HMSA file and XML file match. This is required because HMSA files cannot be decoded without the XML description, and using the wrong XML description could result in corrupted results or undefined software behaviour.
2. To allow software to search for a missing component of the file pair such as a renamed or moved file.

To ensure maximum efficacy of the UID mechanism, software that writes or modifies HMSA files *shall* create new UIDs when:

- Creating a new HMSA/XML dataset.
- Modifying any contents of either the HMSA or XML files.
- Extracting a subset of a HMSA file.

The UID *may* be retained unchanged when:

- Creating an exact copy of a HMSA/XML pair.
- Renaming a HMSA/XML pair.

To further guarantee the integrity of HMSA UIDs, the following is required of UID generation algorithms:

- The output domain of the algorithm *should* span every possible 64-bit value.
- The output *shall not* be a predictable or reproducible sequence of UIDs.

The recommended method of generating a UID is to use a one-way cryptographic hash function, such as the NIST-published SHA-1 algorithm, with a diverse set of inputs to ensure sufficient hash entropy.

2.5 XML Parameter element formats

To maximise compatibility and prevent data misinterpretation, the format of elements and attributes used to store arbitrary parameters in the HMSA XML are strictly defined below.

2.5.1 Numerical data types

The data types of numerical parameters *shall* be explicitly declared using a `DataType` attribute to ensure XML readers can properly load numerical parameters in the appropriate data types without requiring type-guessing code or risking data truncation. The `DataType` attributes are not required for strings of text, or for list elements containing nested elements. The `DataType` attribute, if provided, *shall* take one of the following values:

DataType	Description	Example

"int"	Signed 32 bit integer	`<PointCount DataType="int">`
		`155`
		`</PointCount>`

"int64"	Signed 64 bit integer	`<DataLength DataType="int64">`
		`9223372036854775807`
		`</DataLength>`

"float"	32-bit IEEE 754 single-precision floating point number	`<Gain DataType="float">2.5001</Gain>`

"float64"	64-bit IEEE 754 double-precision floating point number	`<ExampleFloat64 DataType="float64">`
		`1.00`
		`</ExampleFloat64>`

"array:xyz"	An array of values, where ‘xyz’ is one of the above data types	`<Fibonacci DataType="array:int" Count="6">`
		`1, 1, 2, 3, 5, 8`
		`</Fibonacci>`

The values of the `DataType` attributes *shall* be written in lower case.

If the parameter is a member of a dataset template defined in Appendix A, or a condition template defined in Appendix B, the data type *shall* be equal to the type defined in the template.

The "int" and "float" data types *should* be used as the default data types for integral and decimal values, respectively. If greater precision is required for particular condition elements, then the 64-bit versions ("int64", "float64") *may* be used instead, such as with the `<DataOffset>` and `<DataLength>` elements in the dataset definition (see Section 5.2: The `<DataOffset>` and `<DataLength>` elements).

HMSA XML parsers *shall* load parameters using a data type of equal or greater precision to that specified by `DataType` attribute.

If no data type is provided, and the element contains no child elements, HMSA XML parsers *shall* interpret the value to be a text string.

Additional data types are defined for binary data in the HMSA file, as specified in Section 5.3: The `<DatumType>` element. However, parameters in the XML file *shall not* use these additional data types. Only "int", "float", "int64", "float64", and arrays of the same, are permitted data types for parameter elements in the XML file.

2.5.2 Arrays of values

Arrays of values *shall* be specified using a `DataType` attribute of "array:xyz", where xyz is one of the data types specified in Section 2.5.1: Numerical data types. The number of values in the array *shall* be specified using a `Count` attribute, which is assumed to be a decimal text representation of an unsigned 32 bit integer. Array values *shall* be written as comma separated values. For example:

```
<Fibonacci DataType="array:int" Count="6">1, 1, 2, 3, 5, 8</Fibonacci>
```

The value of the `Count` attribute shall be a text representation of an unsigned 32-bit integer with a value of 1...4294967295. The use of the `Count` attribute name is reserved for the purpose of specifying array sizes, and *shall not* be used for other purposes.

2.5.3 Numerical values

Numerical values *shall not* contain digit grouping markers such as commas or spaces.

Text encoding of floating point values *shall* follow the IEEE 754-1985 standard for binary <-> decimal conversion. Furthermore:

- Radix/decimal point marker *shall* be the full stop character (U+002E).
- Exponents *shall* be denoted by either ‘E’ or ‘e’.

2.5.4 Physical units

For numerical values with physical units, the units *should* be defined using a `Unit` attribute. Units *shall* be provided in SI units, SI derived units (e.g. "Pa", "Å"), or one of the customary technique-specific units defined in Appendix C (e.g. "counts", "wt%"). Units *shall* be declared in abbreviated form, with optional single-character SI prefix codes (e.g. "kV", for kilovolt). The list of permitted prefixes is also included in Appendix C.

Dataset and condition objects defined in appendices A and B specify the physical units that must be used for parameters within those objects. The precise formats of the unit text *shall* be consistent with the definitions in the appendices.

To preserve scientific accuracy, it is critical that HMSA files use a consistent scheme for specifying compound units that is readable and writeable by both humans and computers. Aesthetically pleasing representations such as $kg \cdot m \cdot s^{-2}$ are difficult to type and are prone to display or interpretation errors when moving between software packages. To avoid confusion, HMSA files *shall* therefore use only the full stop '.' (U+002E), solidus '/' (U+0047) and numerals 0-9 (U+0030 - U+0039) to represent compound units such as "kg.m/s2". The use of the hyphen-minus sign '-' (U+002D) to indicate negative exponents is permitted only for inverse singular units, such as inverse centimetres (cm-1), but not compound units (e.g. "m/s2", not "m.s-2") . Other methods of superscript markup such as the circumflex accent ^ (U+005E) *shall not* be used. The use of brackets in unit definitions is not permitted.

The Unicode character set defines a number of specific code points for scientific symbols, which are visually identical to non-scientific code points. For example, the Unicode Latin capital A with ring above 'Å' (U+00C5) is visually indistinguishable from the Unicode Ångström symbol 'Å' (U+212B). The casual use of one or the other symbol for the same quantity poses a risk to software compatibility. Consequently, to avoid confusion and maximise compatibility, the lowest code point *shall* be used in cases where a unit symbol could be written in two or more visually indistinguishable characters. Required character substitutions are provided in Appendix D.

When defining concentrations, it is mandatory to specify whether the measurement is molar or atomic (mol%), volumetric (vol%) or mass or weight (wt%). Similarly, when using parts

per million or parts per billion notations for concentration, the nature of the measurement *shall* be specified (e.g. mol_ppm, vol_ppm, wt_ppm.)

2.5.5 Alternative language attributes

In addition to the US English text, values in other languages *may* be specified using `alt-lang-xx[-YY...]` attributes, where ‘xx’ is the language code and ‘YY...’ the locale, as in the form of IETF language tags (i.e. ‘en-US’). For example, the author may be specified as:

```
<Author alt-lang-ru="Фёдор Михайлович Достоевский">
   Fyodor Dostoyevsky
</Author>
```

This method *should* be used only to provide proper nouns in appropriate native languages, such as the names of authors, organisations, or places.

The use of the prefix `alt-lang-` in attribute names is reserved for this purpose and *shall not* be used in other attribute names.

2.5.6 Special characters

In accordance with the XML specification, the following characters *shall not* be used in the names or values of elements or attributes:

- < (U+003C)
- > (U+003E)
- " (U+0022)
- ‘ (U+0027)
- & (U+0026)

When writing XML files, occurrences of these characters in value strings *shall* be converted to their respective XML entities:

- <
- >
- "
- '
- &

Upon loading of XML files, following structural parsing, occurrences of these XML entities in strings *shall* be converted back to their corresponding character values before being presented to users or other software.

2.5.7 Ordering of elements

The order in which elements are listed within the XML file is not specified in general, meaning XML elements may be sorted in any order within their parent XML element unless otherwise specified. A notable example of where the ordering of elements is specified is for the contents of the `<MSAHyperDimensionalDataFile>` document root element, where the child elements *shall* be in the following order: `<Header>`, `<Conditions>`, then `<Data>`. A further example is in the ordering of the `<Dimension>` elements in the `<CollectionDimensions>` and `<DatumDimensions>` lists, where the order of `<Dimension>` elements defines the ordering of data in the binary HMSA file (See Section 6.2: Order of collection dimensions and Section 6.4: Order of datum dimensions). Dataset and condition templates *may* also define a required ordering of elements.

3. The <Header> list element

The <Header> list element contains metadata that principally identifies the title of the document, the author/ownership of the data, and the date/time of collection. Header information *shall not* contain parameters that are required for the interpretation of the experimental data.

3.1 Header items are optional

In keeping with the principle of minimalism (see Section 1.5: Minimalism), all items in the <Header> list element are optional. Some elements, such as the <Checksum>, *should* be included, but are not mandatory. Software that reads HMSA XML files *should not* require the presence of any items in the <Header> list to open, display or process files.

If no items are defined within the <Header> list, the empty header list *shall* be specified as either an empty element (<Header />), or as a conventional matched pair of elements with no contents (<Header></Header>). XML parsers for HMSA XML files *shall* support both styles of empty element declaration.

3.2 The <Checksum> element

The <Header> list *should* include a <Checksum> element to allow software to verify that the binary HMSA file exactly matches that specified in the XML file. The <Checksum> element, if provided, *shall* take the following form:

```
<Checksum Algorithm="SHA-1">
   53AAD59C05D59A40AD746D6928EA6D2D526865FD
</Checksum>
```

The contents of the <Checksum> element *shall* be the hexadecimal-encoded (A-F, 0-9) checksum digest of the entire binary HMSA file. The algorithm used to generate the checksum *shall* be declared using the Algorithm attribute. The checksum algorithm *should* be one of the following algorithms:

- SUM32 (sum of all bytes in the binary HMSA file, truncated to a 32 bit / 8 hexadecimal character value)
- SHA-1 (recommended)

The 'SUM32' algorithm is provided for basic protection against single-bit and some multiple-bit errors, but does not protect against multiple-bit errors with zero sum change. For this reason, the 'SHA-1' algorithm is recommended, as it provides strong detection of any form of modification, and is furthermore a widely supported standard with libraries and implementations available in most programming languages and platforms.

3.3 The <Title>, <Author> and <Owner> elements

The title, author, and legal owner of the document *should* be specified within the <Header> list like so:

```
<Title>Beep Beep</Title>
<Author>Wyle E. Coyote</Author>
<Owner>Acme Inc.</Owner>
```

These elements *may* be provided in languages other than US English using an alternative language attribute alt-lang-xx[-YY] (see Section 2.5.5: Alternative language attributes). For example, the name of the author Leo Tolstoy may be provided in his native Russian Cyrillic script as:

```
<Author alt-lang-ru="Лев Николаéвич Толстóй">Leo Tolstoy</Author>
```

3.4 The <Date>, <Time> and <Timezone> elements

The date and time of the creation of the HMSA file *should* be stored in <Date>, <Time> and <Timezone> elements, of the following format:

```
<Date>1985-10-26</Date>
<Time>20:04:00</Time>
<Timezone>UTC-8 US Pacific Standard Time</Timezone>
```

The <Date> and <Time> values *shall* be written in the ISO 8601 date/time format, with the date as YYYY-MM-DD, and the time as HH:MM:SS in 24 hour format. The <Timezone> value *shall* be given in terms of Universal Coordinated Time as "UTC", "UTC±HH", or "UTC±HH:MM", with the timezone offsets given in hours (HH), hours and minutes (HH:MM), or omitted if the offset is zero. Examples include "UTC", "UTC+10", and "UTC-03:30". Following the timezone UTC offset, the two character ISO 3166-1 alpha-2 country code and full formal timezone name *may* be given, such as in "UTC-4 CA Atlantic Standard Time".

Dates *shall* be encoded according to the Gregorian calendar in the common era (CE / AD).

3.5 The <ArbitraryData> element

Within the binary HMSA file, applications may elect to store blocks of arbitrary and proprietary binary or text data. The location and size of these arbitrary data blocks *should* be declared in the <Header> list element using one or more <ArbitraryData> elements. The <ArbitraryData> element allows compatible applications to find the arbitrary data blocks in the HMSA binary file, and also allows 3rd party applications to preserve unknown arbitrary data blocks when modifying or saving HMSA files. However, blocks of arbitrary data *may* be inserted in HMSA binary files without corresponding <ArbitraryData> declarations.

The <ArbitraryData> XML element *shall* contain <DataOffset> and <DataLength> elements, which respectively define the absolute position and size (in bytes) of the data block in the HMSA binary file, as 64 bit integers. These elements are defined analogously to those used in the dataset definition (See Section 5.2: The <DataOffset> and <DataLength> elements.) A Name attribute *may* be provided to identify the block, and a <Format> element *may* be used to describe the formatting of the arbitrary data block. An example <ArbitraryData> definition is provided below:

```
<ArbitraryData Name="Example Corp. data block #1">
   <DataOffset DataType="int64">176126333</DataOffset>
   <DataLength DataType="int64">3321</DataLength>
   <Format>MAC table</Format>
</ArbitraryData>
```

Additional attributes or XML elements may be specified within the <ArbitraryData> element, but are not defined by this specification.

The first dataset object in a HMSA binary file is present at an offset of 8 bytes (i.e. immediately following the UID, see Section 1.2.1: HMSA general structure), but subsequent datasets may be present at any offset that does not overlap with another dataset (See Section 5.2: The <DataOffset> and <DataLength> elements). Hence, blocks of arbitrary data may be placed in the binary HMSA file after any dataset. This specification places no restrictions on the number or size of arbitrary data blocks present in a HMSA file.

Blocks of arbitrary data in the binary HMSA file *should* commence with a unique identifier or ‘magic number’, so that applications that read arbitrary data from HMSA files may verify the expected formatting of the arbitrary data block. Blocks of arbitrary data in the binary

HMSA file *should not* contain absolute position references to other locations within the file, as 3rd party applications may modify the ordering of datasets and arbitrary data blocks. Relative position references to locations within the same arbitrary data block may be used.

Applications that save or modify HMSA/XML files *may* — but are not required to — preserve blocks of arbitrary data stored in the HMSA binary files. If an application does not preserve arbitrary data blocks when saving HMSA binary files, it *should* remove any `<ArbitraryData>` elements from the `<Header>` list of the XML files. As arbitrary data blocks may be removed from files, applications that read such arbitrary data *should* verify that the expected position of the arbitrary data block lies outside the position range of any declared dataset, and *should* validate any unique identifier or 'magic number' that is given at the start of an arbitrary data block in the HMSA binary file.

3.6 Other optional header elements

The header *may* optionally include any number of other metadata elements, such as:

- `<Client>`
- `<AuthorSoftware>`
- `<Location>`
- `<Comment>`

The formats and conventions of these optional elements are not defined, and these values *shall not* be required for the proper display or interpretation of the experimental data or conditions. Any scientifically meaningful metadata *shall* be stored within an appropriate element within the `<Conditions>` list (See Section 4: The `<Conditions>` list element.)

4. The <Conditions> list element

The <Conditions> element is a list of experimental condition that may assist in the scientific interpretation of the experimental data, such as spectrometer gains and offsets. Conditions are technique-specific, and so there will be a diverse range of possible condition elements. Templates for common conditions are discussed in Section 4.2: Conditions templates and classes, and examples are given in Appendix B.

All condition templates *shall* have the following base structure:

```
<TemplateName Class="ClassName" ID="UniqueStringOfText">
        [...]
</TemplateName>
```

The Class and ID attributes are optional, and may not be present for all elements in the <Conditions> list.

The templates and class names are further described in Section 4.2: Conditions templates and classes, and the ID attribute is described in Section 4.3: Condition identifiers. Note that the <Conditions> list *may* contain any number of entries with the same template name and/or class name. However, the ID attribute, if present, *shall* be unique for each condition entry.

4.1 Conditions are optional

Because of the limitless number of potentially useful condition objects, it is not reasonable to assume that all software must read or understand all condition types. Consequently, HMSA/XML file format has been designed such that all conditions are optional. Software that reads HMSA files *shall* be able to read and display datasets without having to parse and understand any or all of the associated conditions (albeit without calibration or further interpretation.) Conditions therefore *shall not* contain any information that is required to load the dataset from the file, as the position and layout of the dataset object in the HMSA file is completely defined in the relevant dataset object (see Section 5: The `<Data>` list element).

This requirement is intended to ensure a universal base level of support for common dataset types, so that, for example, a program that can read and display *any* 2D rastered spectral map dataset should work with *all* 2D rastered spectral map datasets, from any technique (EELS, XEDS, CL, etc.)

4.2 Conditions templates and classes

The name of the condition object is called the ‘template’. HMSA defines a number of condition templates to accommodate a range of common experimental techniques:

- `<Probe>`, for experimental parameters relating to the instrument’s probe configuration (e.g. beam current, accelerating voltage, etc.)
- `<Detector>`, for experimental parameters relating to the detector configuration (e.g. XEDS, EELS, etc.)
- `<Acquisition>`, for experimental parameters relating to the position and time of one or more measurements of the specimen (e.g. line-scan, map, etc.)

The `Class` attribute is used to define subtypes of condition templates. For instance, the `<Probe>` template supports a class named "EM", which defines general electron column conditions for electron microscopes. This class may be further extended using a subclass, denoted by a solidus ‘/’ (U+002F), such as "EM/TEM" for transmission electron microscopes (which may include lens modes &c).

Each subclass inherits the required and optional parameters of the parent template/class, as well as any restrictions on parameter values. Required parameters *shall not* be removed by subclasses, nor *shall* any restrictions on parameter ranges be violated. Consequently, and object of type `<Acquisition Class="RegularArray/XY">` is both a valid `<Acquisition Class="RegularArray">` object, and a valid `<Acquisition>` object. This class hierarchy system is intended to ensure that software than can interpret a condition object such as an `<Acquisition>` can validly interpret all derived subclasses, even if no additional parameters are read or understood.

To ensure class names are unambiguous and universally typeable, class names *shall* contain only Latin characters and digits from the ASCII subset of the Unicode character set (A-Z, a-z, 0-9), and the hyphen-minus ‘-’ (U+002D). The solidus ‘/’ (U+002F) *shall only* be used to delimit class/subclass names.

A list of supported templates, which is not exhaustive, is provided in Appendix B. It is expected that users of different techniques, or different vendors, may extend these templates/classes to suit their particular needs.

4.3 Condition identifiers

Top-level XML elements in the `<Conditions>` list *may* have a unique identifier string using the `ID` attribute. The purpose of this attribute, in conjunction with the dataset `<IncludeConditions>` list, is to permit disambiguation of multiple condition XML elements with the same template. This may occur in a multi-dataset map, where one condition may apply to one dataset, and another may apply to a second dataset. If the `ID` attribute is specified for a condition element, it *shall not* be shared with any other item in the `<Conditions>` list, regardless of template or class. For maximum compatibility, the `ID` string *should* only contain characters in the Unicode range U+0032 to U+007E, corresponding to printable ASCII characters (excluding special characters, see Section 2.5.6: Special characters).

5. The <Data> list element

The <Data> element is a list of the binary datasets stored in the HMSA file. The <Data> element *shall* contain one or more dataset entries, which describe the address, size, and layout of the binary data within the associated HMSA file. Applications are not required to parse more than the first dataset in the HMSA XML file, but *should* notify the user if additional unparsed datasets are present in the file.

By design, dataset definitions contain no extraneous data that is unrelated to the format of the binary data, such as experimental parameters to assist with the interpretation or display of the data. This arrangement ensures that common dataset types can be used across a range of techniques. For instance, the dataset definition for a spectral map will be identical regardless of whether the dataset was collected via XEDS, CL, EELS, Raman, etc.

By default, it is assumed that all conditions in the <Conditions> list apply to every dataset declared in the <Data> list. Optionally, datasets may explicitly specify a subset of conditions that apply using the <IncludeConditions> list, which may be necessary in multi-dataset files with multiple instances of the same condition template (see Section 5.6: The <IncludeConditions> element).

All dataset templates have the following base structure:

```
<TemplateName Name="Example">
   <DataOffset DataType="int64">123</DataOffset>
   <DataLength DataType="int64">456</DataLength>
   <DatumType>uint16</DatumType>
   <DatumDimensions>
      [ zero or more dimension definitions ]
   </DatumDimensions>
   <CollectionDimensions>
      [ zero or more dimension definitions ]
   </CollectionDimensions>
   <IncludeConditions>
      [ zero or more references to conditions ]
   </IncludeConditions>
</TemplateName>
```

The elements of the base dataset object are defined below.

5.1 Dataset templates and classes

Datasets use the same template/class hierarchy scheme as defined for condition objects in Section 4.2: Condition templates and classes. However, this specification only defines three initial templates, which differ only in the number of collection dimensions. They are:

- `<Single>`, for a measurement of the specimen at a single point, typically but not necessarily a spectrum or image (e.g. diffraction pattern.)
- `<IrregularArray>`, for set of analyses collected in an irregular pattern or sequence. This can be used for an unevenly spaced series of measurements, or sparsely scanned images, for example.
- `<RegularArray>`, for data collected over an *N* dimensional regular grid, where *N* is most commonly 1 for linescans, 2 for X/Y images (including optical micrographs, x-ray maps, etc.), or 3 for X/Y/Z confocal or serial section images.

Examples of dataset templates are provided in Appendix A.

Experimental data that cannot be represented as one of the defined dataset templates *may* be stored in the HMSA binary file as arbitrary data blocks (see Section 3.5: The `<ArbitraryData>` element).

5.2 The <DataOffset> and <DataLength> elements

The location of the beginning of the dataset's binary data within the HMSA file is given in the <DataOffset> element, and is measured in bytes from the start of the file, in 64-bit signed integer precision. The first byte of the file has an offset of 0.

The location of the first dataset in the file *shall* be 8 bytes from the start, meaning there is no padding between the 8-byte UID and the first dataset. The length of the dataset's binary data within the HMSA file is given in the <DataLength> element, and is measured in bytes, in 64-bit integer precision.

If more than one dataset is present in the file, the location of subsequent datasets *shall not* overlap other datasets in the file, and *may* be:

- Non-contiguous. Padding is permitted between datasets, which may be used to store arbitrary or proprietary data that is not defined in this specification (see Section 3.5: The <ArbitraryData> element).
- Out of order. For example, dataset 2 can come after dataset 3 in the HMSA file.

5.3 The <DatumType> element

The data type of an individual numerical measurement within the dataset *shall* be declared using the <DatumType> element, like so:

```
<DatumType>int</DatumType>
```

For spectra and spectral maps, this element declares the data type of a spectrum channel. For images and hyperimage maps, this is the type of an image pixel.

The <DatumType> element *shall* take one of the following values:

DatumType	Size (B)	Description
"byte"	1	Unsigned 8 bit integer
"int16"	2	Signed 16 bit integer
"uint16"	2	Unsigned 16 bit integer
"int"	4	Signed 32 bit integer
"uint"	4	Unsigned 32 bit integer
"int64"	8	Signed 64 bit integer
"float"	4	32-bit IEEE 754 single-precision floating point number
"float64"	8	64-bit IEEE 754 double-precision floating point number

5.4 The <DatumDimensions> element

Dataset datum *may* consist of:

- A single value per datum, such as a pixel in a greyscale image.
- A one dimensional array of values per datum, such as a spectrum per pixel in a hyperspectral map, or three colour elements in an RGB image.
- A two dimensional array of values per datum, such as a full diffraction pattern image at every pixel in a hyperimage map.
- Higher datum dimensionality is permitted, but is not defined in this specification.

The dimensionality and ordering of the datum values is defined in `<DatumDimensions>` element, which *shall* contain zero or more `<Dimension>` elements, as defined below:

5.4.1 The <Dimension> element

Each `<Dimension>` element *shall* define the length of the dimension (e.g. the number of channels in a spectrum), and be of the form:

```
<Dimension DataType="int">1024</Dimension>
```

The data type of the value of the `<Dimension>` element *shall* be explicitly declared using a `DataType` attribute, with the value "int" (a signed 32 bit integer).

5.4.2 Datum as single values

For simple greyscale images, for which there is only a single value per datum (i.e. one value per pixel), the datum dimensionality is zero, and hence the `<DatumDimensions>` element *shall* be empty:

```
<DatumDimensions />
```

or, equivalently:

```
<DatumDimensions></DatumDimensions>
```

5.4.3 Datum as arrays

For datum consisting of a single array of values (e.g. a spectrum per pixel in a spectral map), the datum dimensionality is one, and the `<DatumDimensions>` element *shall* contain one `<Dimension>` element of the form:

```
<DatumDimensions>
  <Dimension DataType="int">1024</Dimension>
</DatumDimensions>
```

Information relating to the identity, calibration and interpretation of the datum dimension *should* be stored in a corresponding `<Detector>` condition element, such as a `<Detector Class="Spectrometer">`.

5.4.4 Datum as 2D arrays

For datum consisting of a 2D array of values (e.g. a diffraction pattern in a hyperimage), the datum dimensionality is two, and the `<DatumDimensions>` element *shall* contain two `<Dimension>` elements of the form:

```
<DatumDimensions>
  <Dimension DataType="int">512</Dimension>
  <Dimension DataType="int">400</Dimension>
</DatumDimensions>
```

The identity, calibration and interpretation of the datum dimensions *should* be defined in a corresponding `<Detector>` condition element, such as a `<Detector Class="Camera">`.

5.4.5 Datum as 3D arrays and higher dimensionality

Higher dimensionality datum (3D, etc.) are supported by the HMSA format, but are not explicitly defined in this specification.

5.5 The <CollectionDimensions> element

The <CollectionDimensions> list element functions analogously to the <DatumDimensions> element (see Section 5.4: The <DatumDimensions> element), and defines the dimensionality and order of the collection of datum across, or though, the specimen. The <CollectionDimensions> list will contain zero or more <Dimension> elements, depending on the type of dataset:

- Zero collection dimensions imply an analysis at a single point, such as the collection of a diffraction pattern at a single position on the specimen, as used with the <Single> dataset template.
- One collection dimension implies either:
    - A regular sequence of analyses, such as a line scan, as used with the <RegularArray> dataset template.
    - An irregular sequence of analyses, such as a random or irregularly rastered map, or a non-periodic time sequence, as used with the <IrregularArray> dataset template.
- Two collection dimensions imply a 2D regular gridded raster, such as an X/Y map, as used with the <RegularArray> dataset template.
- Three collection dimensions imply a 3D regular gridded raster, such as an X/Y/Z map, as used with the <RegularArray> dataset template.

The example below shows the <CollectionDimensions> element for an X/Y/Z 3-dimensional serial section image, as may be stored in a <RegularArray> dataset, where 256 slices were made, and for each slice a 512×400 pixel rastered map was recorded:

```
<CollectionDimensions>
    <Dimension DataType="int32">512</Dimension>
    <Dimension DataType="int32">400</Dimension>
    <Dimension DataType="int32">256</Dimension>
</CollectionDimensions>
```

The ordering of dimensions in the <CollectionDimensions> list *should* be in order from fastest to slowest raster sequence. For example, in the 3D serial section map example above, the X dimension of the rastered image may be the fast scan direction of the microscope, and therefore a full row of pixels in the X dimension are collected before moving to the next coordinate in the Y dimension. Thus, the Y dimension follows after the X dimension in the

<CollectionDimensions> list. Furthermore, since a full image plane of X and Y data are collected before the next section in the Z dimension, the Z dimension follows the X and Y dimensions in the <CollectionDimensions> list. This ordering ensures that data is stored in the files in the order in which it was collected.

The identity of the collection dimensions, and any data relating to calibration of positions, *should* be stored in a corresponding <Acquisition> condition (or subclass thereof).

Examples of dataset templates of different dimensionalities are provided in Appendix A.

5.6 The <IncludeConditions> element

The <IncludeConditions> element is an optional element in the dataset definition that may contain zero or more references to the conditions that should be used to interpret the data in the dataset. If the <IncludeConditions> list is not defined or is empty, all condition specified in the <Conditions> list are assumed to apply to the dataset.

Condition references in the <IncludeConditions>, if used, *shall* take the following form:

```
<ConditionTemplateName>ConditionIdentifier</ConditionTemplateName>
```

...where <ConditionTemplateName> matches the template name for the condition (e.g. <Probe>, <Detector>, etc.). The *ConditionIdentifier* value *shall* match the ID attribute of the element referenced in the <Conditions> list.

For example, to reference a condition defined in the <Conditions> list thusly:

```
<Detector Class = "Spectrometer/XEDS" ID="Windowless SDD">
   [...]
</Detector>
```

...the entry in the dataset's <IncludeConditions> list would be:

```
<Detector>Windowless SDD</Detector>
```

6. Format of datasets in the HMSA binary file

Whilst the HMSA dataset objects supports any number of experimental collection and datum dimensions, binary file streams only have one dimension, that being offset from the start of the file. Therefore, to read and write HMSA binary files a mapping scheme is required to convert from collection and datum coordinates to file offsets.

6.1 Datum-first order

HMSA binary datasets are stored in *datum-first order*, such that the entire measurement data for a given collection coordinate (i.e. pixel in a map) are stored contiguously, followed by the data of the next collection coordinate, and so on. Thus, hyperspectral maps are stored spectrum-by-spectrum, and hyperimage maps (e.g. EBSD pattern maps) are stored image-by-image.

The example below is the dataset definition for a spectral linescan, with one datum dimension (1000 spectrum channels) and one collection dimension (5 linescan positions).

```
<DatumDimensions>
   <Dimension DataType="int">1000</Dimension>
</DatumDimensions>
<CollectionDimensions>
   <Dimension DataType="int">5</Dimension>
</CollectionDimensions>
```

Schematically, this dataset may be represented as a matrix with the datum dimension mapped to the x-axis, and the collection dimension mapped to the y-axis:

0	1	2	3	...	999	Datum 1
1000	1001	1002	1003	...	1999	Datum 2
2000	2001	2002	2003	...	2999	Datum 3
3000	3001	3002	3003	...	3999	Datum 4
4000	4001	4002	4003	...	4999	Datum 5

The numbers in the cells above indicate the order in which the channel values are stored in the dataset in the binary HMSA file, as represented below. Note that with datum-first ordering the complete data for each point (i.e. spectrum) are stored contiguously.

0	1	2	3	...	999	1000	1001	1002	1003	...	1999	...	4000	4001	4002	4003	...	4999
Datum 1						Datum 2						...	Datum 5					

6.2 Order of collection dimensions

For datasets with more than one collection dimension, such as an XY rastered image or an XYZ serial section map, the order of the `<Dimension>` elements declared in the `<CollectionDimensions>` list determines the order in which the data for each collection coordinate is stored in the HMSA binary file. The data of a complete raster of the first collection dimension is stored contiguously, before stepping to the next coordinate in the second collection dimension, and storing another complete raster of the first collection dimension, and so on.

To demonstrate the collection coordinate ordering, consider a 6×4 pixel dataset with no datum dimensions, such as a greyscale image:

```
<DatumDimensions />
<CollectionDimensions>
   <Dimension DataType="int">6</Dimension>
   <Dimension DataType="int">4</Dimension>
</CollectionDimensions>
```

The dataset may be represented are respectively represented as the x- and y-axes of a matrix, as below.

0	1	2	3	4	5	Line 1
6	7	8	9	10	11	Line 2
12	13	14	15	16	17	Line 3
18	19	20	21	22	23	Line 4

In the HMSA binary file, the data of a full line is (e.g. 0-5) is stored in the binary dataset, followed by the next line (e.g. 6-11), and so on:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Line 1						Line 2						Line 3						Line 4					

Notwithstanding the above example, the first, second and third collection dimension in a HMSA dataset do not necessarily correspond to the x-, y- and z-axes of the sample, respectively. Furthermore, the dataset definition does not specify the origin nor positive direction for each collection dimension. The same example 6×4 dataset above may equally be collected or plotted, therefore, with the first collection dimension as the y-axis, and the

second collection dimension as the x-axis, and the origin in the top-right corner of the map. This operation does not change the ordering of the data in the file.

18	12	6	0
19	13	7	1
20	14	8	2
21	15	9	3
22	16	10	4
23	17	11	5
Line 4	Line 3	Line 2	Line 1

The identity, orientation and calibration of the dataset dimensions *should* be defined in an appropriate `<Acquisition>` condition.

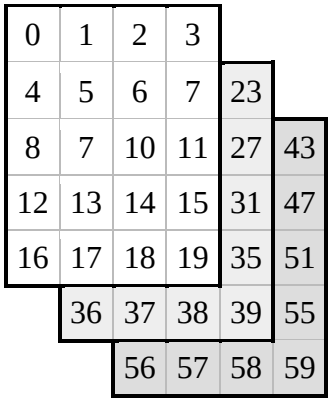
6.3 Higher order collection dimensions

The collection coordinate ordering described above extends trivially to higher dimensions. In a dataset with three collection dimensions, such as a serial-section XYZ map, data is stored as planes of the first two collection dimensions, where each plane is stored as lines of the first collection dimension.

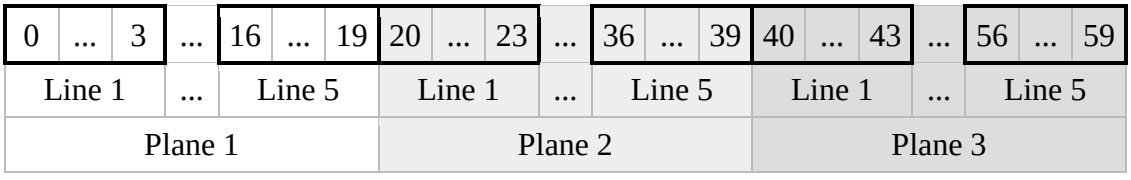
For example, consider a dataset comprising of a 4×5×3 pixel greyscale image from an XYZ serial section dataset. Such a dataset would have three collection dimensions, and 0 datum dimensions, like so:

```
<DatumDimensions />
<CollectionDimensions>
   <Dimension DataType="int">4</Dimension>
   <Dimension DataType="int">5</Dimension>
   <Dimension DataType="int">3</Dimension>
</CollectionDimensions>
```

The 4×5×3 pixel greyscale image may be visualised as a stack of three image planes, each 4×5 pixels in size:



The full data of each image plane (e.g. 0-19) is stored contiguously in the binary dataset, followed by the next plane (e.g. 20-39), and so on, as below. The ordering of pixels within each image plane is the same as for a dataset with two collection dimensions, as described in Section 6.2: Order of collection dimensions.



6.4 Order of datum dimensions

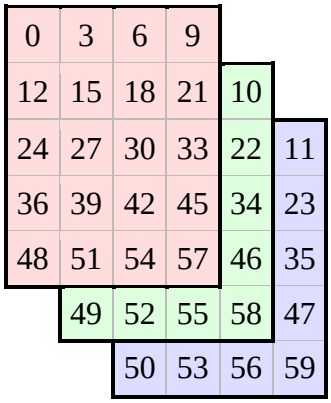
For datasets with more than one datum dimension, such as an EBSD pattern map, the scheme that determines the ordering of datum values in the HMSA binary file is identical to the scheme used for ordering the collection dimensions (See Section 5.7.2: Order of collection dimensions). The order of the `<Dimension>` elements declared in the `<DatumDimensions>` list determines the order in which the data for each datum coordinate is stored in the HMSA binary file. The data of a complete raster of the first datum dimension is stored contiguously, before stepping to the next coordinate in the second datum dimension, and storing another complete raster of the first datum dimension, and so on.

6.5 Hyperspectral map example

To illustrate the ordering of a dataset with collection and datum dimensions that are both non-trivial, consider the example dataset definition below, which is a 4×5 pixel spectral map with three channels per spectrum, such as a red-green-blue image.

```
<DatumDimensions>
   <Dimension DataType="int">3</Dimension>
</DatumDimensions>
<CollectionDimensions>
   <Dimension DataType="int">4</Dimension>
   <Dimension DataType="int">5</Dimension>
</CollectionDimensions>
```

Schematically, this dataset may be represented as a stack of three image planes; one for each datum channel:



The numbers in the cells in the figure above indicate the order in which the values are stored in the dataset in the binary HMSA file, as represented below. Note that with *datum first ordering* the complete data for each pixel (i.e. R, G and B values) are stored contiguously.

0	1	2	3	4	5	6	7	8	9	10	11	...	48	49	50	51	52	53	54	55	56	57	58	59
Pixel 1			Pixel 2			Pixel 3			Pixel 4			...	Pixel 17			Pixel 18			Pixel 19			Pixel 20		
Line 1												...	Line 5											

6.6 Coordinate mapping equations

Following the ordering of collection and datum dimensions defined in sections 6.2 and 6.4, a generalised equation may be defined to determine the location within the HMSA binary file of any measurement datum, which may be useful for out-of-core processing of large datasets, including random seeking. The equation below gives the offset, in bytes from the start of the HMSA binary file, for any measurement datum in a dataset of up to three collection dimensions and up to three datum dimensions. If the datum coordinates are (u, v, w), the collection coordinates are (x, y, z), the datum dimensions sizes are n_u, n_v, and n_w, and the collection dimension sizes are n_x, n_y and n_z, the offset may be expressed as:

$$\text{DatumOffset}(u, v, w, x, y, z) = \text{DataOffset} + \text{DatumSize} \times (u + n_u \times (v + n_v \times (w + n_w \times (x + n_x \times (y + n_y \times z)))))$$

This equation assumes a zero base for all coordinates, such that a dimension d takes values of $0...n_d - 1$. The size in bytes of each individual measurement datum is given by DatumSize, which is determined from the `<DatumType>` element in the dataset definition (see Section 5.3: The `<DatumType>` element). The value of DataOffset, which is the offset relative to the start of the HMSA binary file, is given by the `<DataOffset>` element in the dataset definition (see Section 5.2: The `<DataOffset>` and `<DataLength>` elements).

The offset equation for data of higher dimensionality may be derived by induction by adding additional dimension and coordinate terms. Similarly, the equation may be simplified for data of lower dimensionality by setting the appropriate number of coordinate dimensions (n_u, n_v, n_w, n_x, n_y, or n_z) to 1 and corresponding coordinates (u, v, w, x, y, or z) to zero. Examples for salient dataset types are given below.

Example: Single spectrum, 0 collection dimensions, 1 datum dimension

The equation below gives the offset, in bytes from the start of the HMSA binary file, of channel c in a spectrum of n_c channels.

$$\text{DatumOffset}(c) = \text{DataOffset} + \text{DatumSize} \times c$$

Example: Single diffraction pattern image, 0 collection dimensions, 2 datum dimension

The equation below gives the offset, in bytes from the start of the HMSA binary file, of datum pixel (u, v) in a single image measurement, with $n_u \times n_v$ datum pixels, where u is the first datum dimension, and v is the second.

$$DatumOffset(u, v) = DataOffset + DatumSize \times (u + n_u \times v)$$

Example: Hyperspectral map or colour image, 2 collection dimensions, 1 datum dimension

The equation below gives the offset, in bytes from the start of the HMSA binary file, of channel c in pixel (x, y), in a hyperspectral map of n_c channels, with collection dimensions of $n_x \times n_y$, where x is the first collection dimension and y is the second.

$$DatumOffset(c, x, y) = DataOffset + DatumSize \times (c + n_c \times (x + n_x \times y))$$

Example: Diffraction hyperimage, 2 collection dimensions, 2 datum dimension

The equation below gives the offset, in bytes from the start of the HMSA binary file, of datum pixel (u, v) in collection pixel (x, y), in a hyperimage map of $n_u \times n_v$ datum pixels, with collection dimensions of $n_x \times n_y$, where x is the first collection dimension, and y is the second, and u is the first datum dimension, and v is the second.

$$DatumOffset(u, v, x, y) = DataOffset + DatumSize \times (u + n_u \times (v + n_v \times (x + n_x \times y)))$$

Example: XYZ serial section hyperspectral map, 3 collection dimensions, 1 datum dimension

The equation below gives the offset, in bytes from the start of the HMSA binary file, of channel c in pixel (x, y, z), in a 3D serial section hyperspectral map of n_c channels, with collection dimensions of $n_x \times n_y \times n_z$, where x is the first collection dimension, y is the second, and z is the third.

$$DatumOffset(c, x, y, z) = DataOffset + DatumSize \times (c + n_c \times (x + n_x \times (y + n_y \times z)))$$

Appendix A - Dataset templates and classes

<Single>

The `<Single>` dataset template is used to store a single measurement of a specimen at a single point in space or time. This template does not specify the datum dimensionality.

Restrictions:

The `<CollectionDimensions>` element *shall* contain no entries, like so:

```
<CollectionDimensions />
```

Recommended conditions:

The following conditions *should* be present in the `<Conditions>` list, and referenced in the dataset's `<IncludeConditions>` list (if used):

- `<Acquisition>` (or sub-classes)
- `<Instrument>` (or sub-classes)
- `<Probe>` (or sub-classes)
- `<Detector>` (or sub-classes)
- `<Specimen>` (or sub-classes)

Examples:

XEDS spectrum (0 collection dimensions, 1 datum dimension)

This is an example dataset definition for a single measurement of a 4096 channel XEDS spectrum, stored as unsigned 32-bit integers:

```
<Single Name="Spectrum measurement">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">16384</DataLength>
   <DatumType>uint</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">4096</Dimension>
   </DatumDimensions>
   <CollectionDimensions />
</Single>
```

EBSD diffraction pattern (0 collection dimensions, 2 datum dimensions)

This is an example dataset definition for a single measurement of a 1024×800 pixel diffraction pattern, stored as unsigned 16-bit integers:

```
<Single Name="Pattern measurement">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">1638400</DataLength>
   <DatumType>uint16</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">1024</Dimension>
      <Dimension DataType="int">800</Dimension>
   </DatumDimensions>
   <CollectionDimensions />
</Single>
```

Note this dataset type *shall not* be used to store two dimensional images rastered over the specimen, such as a conventional TEM or SEM image. Instead, such data *shall* be stored using the `<RegularArray>` dataset template with two collection dimensions.

<IrregularArray>

The `<IrregularArray>` dataset template represents a sequence of point measurements collected under the same conditions but in an irregular pattern, such as a line scan, a time sequence, or sparsely scanned images. The data in the HMSA file is stored analysis-by-analysis, without padding. This template does not specify the datum dimensionality.

Restrictions:

The `<CollectionDimensions>` element *shall* contain exactly one `<Dimension>` item, like so:

```
<CollectionDimensions>
   <Dimension DataType="int">12568</Dimension>
</CollectionDimensions>
```

Recommended conditions:

The following conditions *should* be present in the `<Conditions>` list, and referenced in the dataset's `<IncludeConditions>` list (if used):

- `<Acquisition Class="IrregularArray">`, or equivalent (or sub-classes)
- `<Instrument>` (or sub-classes)
- `<Probe>` (or sub-classes)
- `<Detector>` (or sub-classes)
- `<Specimen>` (or sub-classes)

Examples:

Sequence of XEDS point analyses (1 collection dimension, 1 datum dimension)

This is an example dataset definition for a sequence of XEDS spectrum acquisitions, in which 47 measurements were taken, with each measurement being a 4096 channel spectrum, stored as unsigned 32-bit integers ("uint"):

```
< IrregularArray Name="Example XEDS sequence">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">770048</DataLength>
   <DatumType>uint</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">4096</Dimension>
   </DatumDimensions>
   <CollectionDimensions>
      <Dimension DataType="int">47</Dimension>
   </CollectionDimensions>
</IrregularArray>
```

<RegularArray>

The `<RegularArray>` dataset template represents a dataset that has been rastered over regularly spaced intervals in one or more dimensions, such as a 1D linescan, a 2D image, or a 3D serial section. This template does not specify the datum dimensionality.

Restrictions:

The `<CollectionDimensions>` list *shall* contain one or more `<Dimension>` elements, which *shall* be of the form:

```
<Dimension DataType="int">314159</Dimension>
```

Recommended conditions:

The following conditions *should* be present in the `<Conditions>` list, and referenced in the dataset's `<IncludeConditions>` list (if used):

- `<Acquisition Class="RegularArray">` (or sub-classes)
- `<Instrument>` (or sub-classes)
- `<Probe>` (or sub-classes)
- `<Detector>` (or sub-classes)
- `<Specimen>` (or sub-classes)

Examples:

EELS elemental linescan (1 collection dimension, 0 datum dimensions)

This is an example dataset definition for an EELS elemental linescan, in which an evenly-stepped sequence of 128 measurements were taken, with each measurement being a single value for the background-subtracted intensity of an element of interest, stored as a double-precision float ("float64"):

```
<RegularArray Name="Example EELS elemental linescan">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">1024</DataLength>
   <DatumType>float64</DatumType>
   <DatumDimensions />
   <CollectionDimensions>
      <Dimension DataType="int">128</Dimension>
   </CollectionDimensions>
</RegularArray>
```

XEDS spectral linescan (1 collection dimension, 1 datum dimension)

This is an example dataset definition for an XEDS spectral linescan, in which an evenly-stepped sequence of 512 spectra were recorded, with each spectrum consisting of 4096 channels, and each channel is a unsigned 16-bit integer ("uint16"):

```
<RegularArray Name="Example XEDS linescan">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">4194304</DataLength>
   <DatumType>uint16</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">4096</Dimension>
   </DatumDimensions>
   <CollectionDimensions>
      <Dimension DataType="int">512</Dimension>
   </CollectionDimensions>
</RegularArray>
```

Backscattered electron image (2 collection dimensions, 0 datum dimensions)

This is an example dataset definition for a backscattered electron image, in which a raster grid of 512×400 pixel measurements were taken of the backscatter detector output, stored as an unsigned short integer ("uint16"):

```
<RegularArray Name="Example BSE image">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">409600</DataLength>
   <DatumType>uint16</DatumType>
   <DatumDimensions />
   <CollectionDimensions>
      <Dimension DataType="int">512</Dimension>
      <Dimension DataType="int">400</Dimension>
   </CollectionDimensions>
</RegularArray>
```

Colour optical micrograph (2 collection dimensions, 1 datum dimension)

This is an example dataset definition for a colour optical micrograph, in which an 5184×3456 image was recorded, with three colour channels at each pixel (e.g. RGB), and each colour stored as a single byte:

```
<RegularArray Name="Example 17.9Mpx optical image">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">53747712</DataLength>
   <DatumType>byte</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">3</Dimension>
   </DatumDimensions>
   <CollectionDimensions>
      <Dimension DataType="int">5184</Dimension>
      <Dimension DataType="int">3456</Dimension>
   </CollectionDimensions>
</RegularArray>
```

Cathodoluminescence spectral map (2 collection dimensions, 1 datum dimension)

This is an example dataset definition for a spectral cathodoluminescence map, in which an raster grid of 4000×3000 pixel spectra were measured, with 1024 channels per spectrum, and each channel is a 32-bit floating point number ("float"):

```
<RegularArray Name="Example CL spectral map image">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">49152000000</DataLength>
   <DatumType>float</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">1024</Dimension>
   </DatumDimensions>
   <CollectionDimensions>
      <Dimension DataType="int">4000</Dimension>
      <Dimension DataType="int">3000</Dimension>
   </CollectionDimensions>
</RegularArray>
```

3D serial section EBSD pattern map (3 collection dimensions, 2 datum dimensions)

This is an example dataset definition for a serial-section EBSD map, where:

- 300 vertical sections were made,
- a 2048×1024 EBSD map was performed on each section,
- a 512×400 pixel diffraction pattern was recorded at each pixel in each EBSD map, and;
- each pixel in the diffraction pattern is an unsigned short integer ("uint16"):

```
<RegularArray Name="Example 3D serial section EBSD pattern map">
   <DataOffset DataType="int64">8</DataOffset>
   <DataLength DataType="int64">
      257698037760000
   </DataLength>
   <DatumType>uint16</DatumType>
   <DatumDimensions>
      <Dimension DataType="int">512</Dimension>
      <Dimension DataType="int">400</Dimension>
   </DatumDimensions>
   <CollectionDimensions>
      <Dimension DataType="int">2048</Dimension>
      <Dimension DataType="int">1024</Dimension>
      <Dimension DataType="int">300</Dimension>
   </CollectionDimensions>
</RegularArray>
```

Appendix B - Condition templates and classes

<Acquisition>

The `<Acquisition>` condition template is a generic object that describes the position and duration of one or more measurements of the specimen. This template *should not* be used directly. Instead, use a sub-class appropriate for the type of acquisition, such as:

- `<Acquisition Class="Single">`,
- `<Acquisition Class="IrregularArray">`,
- `<Acquisition Class="RegularArray">`,
- `<Acquisition Class="RegularArray/Linescan">`,
- `<Acquisition Class="RegularArray/XY">`, or;
- `<Acquisition Class="RegularArray/XY/Z">`.

Optional elements:

The `<DateTime>` element

The date and time of the start of the acquisition *should* be recorded in a `<DateTime>` element (see `<DateTime>` condition template), like so:

```
<DateTime>
  <Date>1985-10-26</Date>
  <Time>20:04:00</Time>
  <Timezone>UTC+10</Timezone>
</DateTime>
```

The <SpecimenPosition> element

The coordinates of the acquisition on the specimen *should* be provided using a <SpecimenPosition> element (See <SpecimenPosition> condition template). If defined, the <SpecimenPosition> element *shall* take the form:

```
<SpecimenPosition>
   <X Unit="mm" DataType="float">1.0</X>
   <Y Unit="mm" DataType="float">-5.0</Y>
   <Z Unit="mm" DataType="float">10.0</Z>
   <R Unit="degrees" DataType="float">90.0</R>
   <EulerRotation Sequence="x" Unit="degrees" DataType="float">
      -70
   </EulerRotation>
</SpecimenPosition>
```

The interpretation of the <SpecimenPosition> varies depending on the <Acquisition> sub-class:

- For <Acquisition Class="Single">, the <SpecimenPosition> element defines the position of the single analysis.
- For <Acquisition Class="RegularArray"> , the <SpecimenPosition> element defines the location of the mid-point or starting coordinate of the regular array, depending on the value of the Name attribute.
- For <Acquisition Class="IrregularArray">, multiple <SpecimenPosition> elements are included within a <SpecimenPositionList> element to explicitly define the position of each analysis.

The <TotalTime> element

If the acquisition includes multiple measurements (such as a linescan or map), the <TotalTime> element may be used to define the total *real time* taken to collect all measurements in the acquisition set. If provided, the <TotalTime> element *shall* be of the form:

```
<TotalTime Unit="s" DataType="float">14400.0</TotalTime>
```

The <DwellTime> element

The <DwellTime> element may be used to define the uniform *real time* taken for each individual measurement, such as a point spectrum acquisition, a single point in a linescan, or a pixel in a map. If provided, the <DwellTime> element *shall* be of the form:

```
<DwellTime Unit="s" DataType="float">35.0</DwellTime>
```

The <DwellTime_Live> element

The <DwellTime_Live> element may be used to define the detector *live time* for each individual measurement, if known. If provided, the <DwellTime_Live> element *shall* be of the form:

```
<DwellTime_Live Unit="s" DataType="float">35.0</DwellTime_Live>
```

Example:

Examples of the <Acquisition> condition template are provided for sub-classes, including:

- <Acquisition Class="Single">,
- <Acquisition Class="IrregularArray">,
- <Acquisition Class="RegularArray">,
- <Acquisition Class="RegularArray/Linescan">,
- <Acquisition Class="RegularArray/XY">, and;
- <Acquisition Class="RegularArray/XY/Z">.

<Acquisition Class="Single">

The <Acquisition Class="Single"> condition template defines the position and duration for a singular measurement of the specimen, such as may be used with a <Single> dataset. The <Acquisition Class="Single"> template does not define any additional elements to those of the base <Acquisition> template.

Base template:

- <Acquisition>

Example:

The example below shows the usage of <Acquisition Class="Single"> for a single spectrum acquisition.

```
<Acquisition Class="Single">
   <DwellTime Unit="s" DataType="float">30.0</DwellTime>
   <DwellTime_Live Unit="s" DataType="float">25.0</DwellTime_Live>
   <DateTime>
      <Date>2014-03-07</Date>
      <Time>16:18:06</Time>
      <Timezone>UTC+11 AUS Eastern Daylight Time</Timezone>
   </DateTime>
   <SpecimenPosition>
      <X Unit="mm" DataType="float">1.0</X>
      <Y Unit="mm" DataType="float">-5.0</Y>
      <Z Unit="mm" DataType="float">10.0</Z>
   </SpecimenPosition>
<Acquisition Class="Single">
```

<Acquisition Class="IrregularArray">

The <Acquisition Class="IrregularArray"> condition template defines the position and duration of an irregular sequence of measurements of the specimen, such as may be used with a <IrregularArray> dataset (e.g. sparsely scanned map, sporadic time sequence, etc.)

Base template:

- <Acquisition>

Optional elements:

The <SpecimenPositionList> element

If the <SpecimenPositionList> list element may be used to define the physical location on the specimen of each analysis in an <IrregularArray> dataset. If provided, the number of <SpecimenPosition> elements within the list *shall* be equal to the value of the <CollectionDimension> in the associated <IrregularArray> dataset. The <SpecimenPositionList> definition *shall* be of the form:

```
<SpecimenPositionList>
   <SpecimenPosition>...</SpecimenPosition>
   <SpecimenPosition>...</SpecimenPosition>
   [...]
</SpecimenPositionList>
```

If the <X>, <Y>, <Z>, <R> and/or <EulerRotation> values do not change between successive <SpecimenPosition> elements, the invariant coordinates may be omitted from subsequent <SpecimenPosition> entries.

The <DateTimeList> element

If the <DateTimeList> list element may be used to define the time of each analysis in an <IrregularArray> dataset. If provided, the number of <DateTime> elements within the list *shall* be equal to the value of the <CollectionDimension> in the associated <IrregularArray> dataset. The <DateTimeList> definition *shall* be of the form:

```
<DateTimeList>
   <DateTime>...</DateTime>
   <DateTime>...</DateTime>
   [...]
</DateTimeList>
```

The <Date> and <Timezone> elements *should* be declared in the first <DateTime> element in the <DateTimeList>. Thereafter, the <Date> and <Timezone> elements may be omitted from subsequent <DateTime> entries if their values are unchanged from the preceding entry. The example below shows a sequence of four <DateTime> elements, where the date rolls over after the second element:

```
<DateTimeList>
   <DateTime>
      <Date>2015-12-31</Date>
      <Time>23:59:58</Time>
      <Timezone>UTC+11 AUS Eastern Daylight Time</Timezone>
   </DateTime>
   <DateTime>
      <Time>23:59:59</Time>
   </DateTime>
   <DateTime>
      <Date>2016-01-01</Date>
      <Time>00:00:00</Time>
   </DateTime>
   <DateTime>
      <Time>00:00:01</Time>
   </DateTime>
</DateTimeList>
```

The <DwellTime> and <DwellTime_Live> elements

If the real dwell time per measurement is variable in the irregular array dataset, the <DwellTime> element *should* be defined as an array of values. Likewise, if the live time per measurement is variable in the irregular array dataset, the <DwellTime_Live> element *should* be defined as an array of values. The number of values in the arrays *shall* be equal to the value of the <CollectionDimension> in the associated <IrregularArray> dataset. The <DwellTime> and <DwellTime_Live> arrays shall be of the form:

```
<DwellTime Unit="s" DataType="array:float" Count="4">
   30, 60, 90, 120
</DwellTime>
<DwellTime_Live Unit="s" DataType="array:float" Count="4">
   20, 40, 60, 80
</DwellTime_Live>
```

Example:

The following example shows the use of the `<Acquisition Class="IrregularArray">` condition for three point analyses, where the X and then Y coordinates are changed between measurements, and the dwell time is different for each measurement:

```
<Acquisition Class="IrregularArray">
   <TotalTime Unit="s" DataType="float">
      153
   </TotalTime>
   <DwellTime Unit="s" DataType="array:float" Count="3">
      30, 60, 90
   </DwellTime>
   <DwellTime_Live Unit="s" DataType="array:float" Count="3">
      20, 40, 60
   </DwellTime_Live>
   <SpecimenPositionList>
      <SpecimenPosition>
         <X Unit="mm" DataType="float">1.0</X>
         <Y Unit="mm" DataType="float">-5.0</Y>
         <Z Unit="mm" DataType="float">10.0</Z>
      </SpecimenPosition>
      <SpecimenPosition>
         <X Unit="mm" DataType="float">1.05</X>
      </SpecimenPosition>
      <SpecimenPosition>
         <Y Unit="mm" DataType="float">-5.3</Y>
      </SpecimenPosition>
   </SpecimenPositionList>
   <DateTimeList>
      <DateTime>
         <Date>2014-03-07</Date>
         <Time>16:18:06</Time>
         <Timezone>UTC+11 AUS Eastern Daylight Time</Timezone>
      </DateTime>
      <DateTime>
         <Time>16:18:37</Time>
      </DateTime>
      <DateTime>
         <Time>16:19:38</Time>
      </DateTime>
   </DateTimeList>
</Acquisition>
```

<Acquisition Class="RegularArray">

The `<Acquisition Class="RegularArray">` condition template is a generic object that defines the position and duration of a regular raster over the specimen. This template *should not* be used directly. Instead, use a sub-class appropriate for the type of raster, such as:

- `<Acquisition Class="Raster/Linescan">`,
- `<Acquisition Class="RegularArray/XY">`, or;
- `<Acquisition Class="RegularArray/XY/Z">`.

Base template:

- `<Acquisition>`

Optional elements:

The `<SpecimenPosition>` element

If the `<SpecimenPosition>` element is defined (see the `<Acquisition>` base class), a `Name` attribute *shall* be declared, and take one of the following values:

- "Origin", indicating the position values define the specimen coordinates of the first location in all collection dimensions, or;
- "Center", indicating the position values define the specimen coordinates of the mid-point in all collection dimensions.

If defined, the `<SpecimenPosition>` element *shall* take the general form:

```
<SpecimenPosition Name="Origin">
   <EulerRotation Sequence="x" Unit="degrees" DataType="float">
      -70
   </EulerRotation>
   <X Unit="mm" DataType="float">1.0</X>
   <Y Unit="mm" DataType="float">-5.0</Y>
   <Z Unit="mm" DataType="float">10.0</Z>
   <R Unit="degrees" DataType="float">90.0</R>
</SpecimenPosition>
```

Please refer to the definition of the <SpecimenPosition> condition for a description of the ordering and interpretation of the component coordinates.

The <StepSize> and <StepSizeList> elements

If the step sizes in all the collection dimensions of the associated dataset are equal, or the dataset only contains a single collection dimension, then the step size *should* be specified as a single value using the <StepSize> element:

```
<StepSize Unit="um" DataType="float">1.0</StepSize>
```

However, if the dataset contains more than one collection dimension, *and* if the step size in those collection dimensions are not equal, then the step size *shall* be specified as multiple <StepSize> child elements within a <StepSizeList> element. The number <StepSize> elements *shall* be equal to the number of collection dimensions in the associated dataset, and the order of values *shall* be the same as the order of the collection dimensions specified in the dataset (See Section 5.5: The <CollectionDimensions> element). An example for an XYZ rastered map is given below, where the step size in X and Y were equal (1μm), and the step size in Z is different (5mm).

```
<StepSizeList>
   <StepSize Unit="um" DataType="float">1.0</StepSize>
   <StepSize Unit="um" DataType="float">1.0</StepSize>
   <StepSize Unit="mm" DataType="float">5.0</StepSize>
</StepSizeList>
```

The `<CollectionAxes>` element

By default, the collection dimensions of the dataset are assumed to be the X, Y and Z axes of the specimen coordinate system (See `<SpecimenPosition>` condition template for the definition of the coordinate system). However, some instruments may raster in directions oblique to the X, Y and Z axes of the specimen coordinate system. For instance, a linescan need not follow the X or Y axes, and an XY map may not be rastered in X then Y. The axis vectors for the direction of each collection dimension, relative to the specimen coordinate system, may be explicitly defined using the `<CollectionAxes>` element, like so:

```
<CollectionAxes>
   <Axis DataType="array:float" Count="3">1, 0, 0</Axis>
   <Axis DataType="array:float" Count="3">0, 1, 0</Axis>
</CollectionAxes>
```

If the `<CollectionAxes>` element is defined, the number of `<Axis>` elements *shall* be equal to the number of collection dimensions in the associated dataset definition. The number of vector components of the `<Axis>` elements *shall* be 3, representing the specimen X, Y and Z axes, in that order. Each vector *should* be normalised to a length of 1. Negative values for vector components may be used, indicating the collection dimensions stepped in the negative X, Y and/or Z directions. The collection axis vectors may be non-orthogonal.

Example:

Examples of the `<Acquisition Class="RegularArray">` condition template are provided for sub-classes, including:

- `<Acquisition Class="RegularArray/Linescan">`,
- `<Acquisition Class="RegularArray/XY">`, and;
- `<Acquisition Class="RegularArray/XY/Z">`.

<Acquisition Class="RegularArray/Linescan">

The `<Acquisition Class="RegularArray/Linescan">` condition template defines the position and duration of a one-dimensional raster over the specimen, such as may be used with a `<RegularArray>` dataset with one collection dimension. This template applies only to a linear sequence of steps, using equal spatial step sizes and constant dwell times for each measurement. For irregular step sizes, refer to the `<Acquisition Class="IrregularArray">` template.

Base templates:

- `<Acquisition Class="RegularArray">`
- `<Acquisition>`

Optional elements:

The `<RasterMode>` element

The raster method by which the dataset is collected may be recorded using the `<RasterMode>` element, as below:

```
<RasterMode>Specimen</RasterMode>
```

Recognised values include

- "Probe", for when the probe is rastered over a fixed specimen (e.g. beam linescan in an STEM), or;
- "Specimen", for when the specimen is rastered under a fixed probe (e.g. stage linescan in an EPMA).

The <FrameCount> element

If the linescan is performed by integrating multiple passes over the same points, the number of passes *should* be defined using the <FrameCount> element, as below.

```
<FrameCount DataType="int">40</FrameCount>
```

Note that if the number of frames is greater than 1, the <DwellTime> element of the base <Acquisition> condition refers to the total integrated time for each point, not the time for each point in each pass.

Example:

The example below is for a spectral linescan, in which five 60s spectra were measured at a spacing of 10 microns. The linescan direction (as defined by the CollectionAxes element) is 45 degrees to the rear right of the instrument. The live time is explicitly recorded for each spectrum acquisition using the <DwellTime_Live> element.

```
<Acquisition Class="RegularArray/Linescan">
   <TotalTime Unit="s" DataType="float">304</TotalTime>
   <DwellTime Unit="s" DataType="float">60</DwellTime>
   <DwellTime_Live Unit="s" DataType="array:float" Count="5">
      48, 55, 52, 59, 51
   </DwellTime_Live>
   <DateTime>
      <Date>2014-03-07</Date>
      <Time>16:18:06</Time>
      <Timezone>UTC+11</Timezone>
   </DateTime>
   <StepSize Unit="um" DataType="float">10.0</StepSize>
   <SpecimenPosition Name="Origin">
      <X Unit="mm" DataType="float">1.0</X>
      <Y Unit="mm" DataType="float">-5.0</Y>
      <Z Unit="mm" DataType="float">10.0</Z>
   </SpecimenPosition>
   <CollectionAxes>
      <Axis DataType="array:float" Count="3">1, -1, 0</Axis>
   </CollectionAxes>
</Acquisition>
```

<Acquisition Class="RegularArray/XY">

The <Acquisition Class="RegularArray/XY"> condition template defines the position and duration of a two-dimensional X/Y raster over the specimen, such as may be used with a <RegularArray> dataset with two collection dimensions.

Base templates:

- <Acquisition Class="RegularArray">
- <Acquisition>

Optional elements:

The <RasterMode> element

The raster method by which the dataset is collected may be recorded using the <RasterMode> element, as below:

```
<RasterMode>Specimen</RasterMode>
```

Recognised values include:

- "Probe", for when the probe is rastered over a fixed specimen (e.g. beam map in an STEM), or;
- "Specimen", for when the specimen is rastered under a fixed probe (e.g. stage map in an EPMA).

The <FrameCount> element

If the X/Y raster is performed by integrating multiple passes over the same points, the number of passes *should* be defined using the <FrameCount> element, as below.

```
<FrameCount DataType="int">40</FrameCount>
```

Note that if the number of frames is greater than 1, the <DwellTime> element of the base <Acquisition> condition refers to the total integrated time for each point, not the time for each point in each pass.

Example:

The example below is for a 2 dimensional stage map, using a 300nm step size for both axes, and a 20ms dwell time (in real/clock time). Unusually, the `<CollectionAxes>` define the fast raster direction of the map is the positive Y axis (towards the front of the instrument), and the slow raster direction is in the negative X axis (to the left of the instrument).

```
<Acquisition Class="RegularArray/XY">
   <TotalTime Unit="s" DataType="float">52428.8</TotalTime>
   <DwellTime Unit="ms" DataType="float">20</DwellTime>
   <DateTime>
      <Date>2014-03-07</Date>
      <Time>16:18:06</Time>
      <Timezone>UTC+11</Timezone>
   </DateTime>
   <RasterMode>Specimen</RasterMode>
   <StepSize Unit="um" DataType="float">0.3</StepSize>
   <SpecimenPosition Name="Origin">
      <X Unit="mm" DataType="float">1.0</X>
      <Y Unit="mm" DataType="float">-5.0</Y>
      <Z Unit="mm" DataType="float">10.0</Z>
   </SpecimenPosition>
   <CollectionAxes>
      <Axis DataType="array:float" Count="3">
            0, 1, 0
      </Axis>
      <Axis DataType="array:float" Count="3">
            -1, 0, 0
      </Axis>
   </CollectionAxes>
</Acquisition>
```

<Acquisition Class="RegularArray/XY/Z">

The <Acquisition Class="RegularArray/XY/Z"> condition template is an extension to the <Acquisition Class="RegularArray/XY"> template that supports multiple XY slices at different points in the Z direction, such as may be used with a <RegularArray> dataset with three collection dimensions.

Base templates:

- <Acquisition Class="RegularArray/XY">
- <Acquisition Class="RegularArray">
- <Acquisition>

Optional elements:

The <ZRasterMode> element

The raster method by which the Z-axis is stepped *should* be recorded using the <ZRasterMode> element, as below:

```
<ZRasterMode>Confocal</ZRasterMode>
```

The value of <ZRasterMode> is technique dependent, but may include "Confocal", "FIB", "Mechanical polish", etc.

Example:

The example below is for a 3 dimensional map, in which the X and Y axes are probe scanned, and the Z axis was collected by serial FIB section. The X and Y step size are 500nm, and the Z step size is 10μm.

```
<Acquisition Class="RegularArray/XY/Z">
   <TotalTime Unit="s" DataType="float">524288.</TotalTime>
   <DwellTime Unit="ms" DataType="float">20</DwellTime>
   <DateTime>
      <Date>2014-03-07</Date>
      <Time>16:18:06</Time>
      <Timezone>UTC</Timezone>
   </DateTime>
   <RasterMode>Probe</RasterMode>
   <ZRasterMode>FIB</ZRasterMode>
   <StepSizeList>
      <StepSize Unit="nm" DataType="float">500</StepSize>
      <StepSize Unit="nm" DataType="float">500</StepSize>
      <StepSize Unit="um" DataType="float">10</StepSize>
   </StepSize>
   <SpecimenPosition Name="Center">
      <X Unit="mm" DataType="float">0.0</X>
      <Y Unit="mm" DataType="float">0.0</Y>
      <Z Unit="mm" DataType="float">500.0</Z>
   </SpecimenPosition>
</Acquisition>
```

<Calibration>

The <Calibration> condition template describes the calibration of a set of measurement ordinals with respect to a physical quantity, such as converting channels in an EELS spectrum to energy, or steps in a WDS peak scan to position, angle, wavelength or energy. This template *should not* be used directly. Instead, use a sub-class appropriate for the calibration relationship, such as:

- <Calibration Class="Constant">,
- <Calibration Class="Linear">,
- <Calibration Class="Polynomial">, or;
- <Calibration Class="Explicit">.

Note that calibration condition objects are embedded within other conditions, such as the <Detector Class = "Spectrometer"> condition template.

Required elements:

The <Quantity> element

The physical quantity of the calibration object *shall* be defined using the <Quantity> element, like so:

```
<Quantity>Energy</Quantity>
```

This specification does not enumerate or restrict the values of the <Quantity> element. However, the following examples values are provided:

- Energy
- Wavelength
- Wavenumber
- Position
- Depth

The <Unit> element

The physical element of the calibration unit *shall* be defined by the <Unit> element, like so:

```
<Unit>eV</Unit>
```

The list of permitted values of the <Unit> element is defined in Appendix C - Units and prefixes.

<Calibration Class="Constant">

The `<Calibration Class="Constant">` element defines the energy/wavelength/etc calibration of a spectrometer or other measurement device operating at a fixed position, such as a CL monochromator.

Base template:

- `<Calibration>`

Required elements:

The `<Value>` element

The single calibration value *shall* be defined using the `<Value>` element, like so:

```
<Value DataType="float">-237.098251</Value>
```

The `<Value>` element *shall not* include a declaration of physical units using a `Unit` attribute. Instead, the physical unit is declared in the separate `<Unit>` element, as required by the base `<Calibration>` template.

Example:

```
<Calibration Class="Constant">
  <Quantity>Energy</Quantity>
  <Unit>eV</Unit>
  <Value DataType="float">-237.098251</Value>
</Calibration>
```

<Calibration Class="Linear">

The `<Calibration Class="Linear">` element defines the energy/wavelength/etc calibration of a spectrometer or other measurement device, for which the measurement ordinals (e.g. channel numbers) have a linear relationship to the physical quantity (e.g. nm), with a constant offset and gain.

Base template:

- `<Calibration>`

Required elements:

The `<Offset>` element

The `<Offset>` element *shall* define the calibration value (energy, wavelength, position, etc.) corresponding to the first measurement ordinal. For example, with an XEDS detector, the value of the `<Offset>` element is the energy of channel 0.

```
<Offset DataType="float">-237.098251</Offset>
```

The `<Gain>` element

The `<Gain>` element *shall* define the linear increment in calibration values from one ordinal to the next. For example, with an XEDS detector, the value of the `<Gain>` element is the number of electron volts per channel.

```
<Gain DataType="float">2.49985</Gain>
```

The `<Gain>` and `<Offset>` elements *shall not* include declarations of physical units using a `Unit` attribute. Instead, the physical unit is declared for both parameters in the separate `<Unit>` element, as required by the base `<Calibration>` template.

Example:

```
<Calibration Class="Linear">
  <Quantity>Energy</Quantity>
  <Unit>eV</Unit>
  <Gain DataType="float">2.49985</Gain>
  <Offset DataType="float">-237.098251</Offset>
</Calibration>
```

<Calibration Class="Polynomial">

The `<Calibration Class="Polynomial">` element defines the energy/wavelength/etc calibration of a spectrometer or other measurement device, for which the measurement ordinals (e.g. channel numbers) have a relationship to the physical quantity (e.g. nm) that may be modelled by an n^{th} order polynomial.

Base template:

- `<Calibration>`

Required elements:

The `<Coefficients>` element

The calibration coefficients *shall* be defined using the `<Coefficients>` element, as an array of floating point values (see Section 2.5.2: Arrays of values)

```
<Coefficients DataType="array:float" Count="4">
   -2.225, 0.677, 0.134, -0.018
</Coefficients>
```

The `<Coefficients>` element *shall not* include declarations of physical units using a `Unit` attribute. Instead, the physical unit is declared for all in the separate `<Unit>` element, as required by the base `<Calibration>` template.

Example:

```
<Calibration Class="Polynomial">
   <Quantity>Energy</Quantity>
   <Unit>eV</Unit>
   <Coefficients DataType="array:float" Count="4">
      -2.225, 0.677, 0.134, -0.018
   </Coefficients>
</Calibration>
```

<Calibration Class="Explicit">

The `<Calibration Class="Explicit">` element defines the energy/wavelength/etc calibration of a spectrometer or other measurement device, for which relationship between the measurement ordinals (e.g. channel numbers) and physical quantity (e.g. nm) cannot be adequately modelled by linear or polynomial functions, and therefore must be declared explicitly for each ordinal.

Base template:

- `<Calibration>`

Required elements:

The `<Values>` element

The explicit range of physical values for the calibration object *shall* be defined using the `<Values>` element, as an array of floating point values (see Section 2.5.2: Arrays of values), like so:

```
<Values DataType="array:float" Count="1024">
   198.557114, 199.364639, 200.172089, 200.979446, 201.786743, 202.593948,
...
</Values>
```

The `<Values>` element *shall not* include declarations of physical units using a `Unit` attribute. Instead, the physical unit is declared in the separate `<Unit>` element, as required by the base `<Calibration>` template.

Optional elements:

The `<Labels>` element

The `<Labels>` element may be used to provide a list of comma separated values (CSV) for text labels of each of the calibration points. If used, the number of CSV text strings in the `<Labels>` element *shall* be equal to the number of items in `<Values>` array.

```
<Labels>Peak, BgndLow, BgndHigh</Labels>
```

Example:

```
<Calibration Class="Explicit">
  <Quantity>Wavelength</Quantity>
  <Unit>nm</Unit>
  <Values DataType="array:float" Count="1024">
    198.557114, 199.364639, 200.172089, 200.979446, 201.786743, 202.593948,
...
  </Values>
</Calibration>
```

<Composition>

The `<Composition>` conditions template is a generic object that describes the composition of a material. This template should not be used directly; instead use a sub-class such as `<Composition Class = "Elemental">`.

The general form of `<Composition>` condition is a list of components (such as elements, compounds, isotopes, etc.), like so:

```
<Composition Class = "Elemental" Unit="wt%">
   [...one or more components...]
</Composition>
```

The formatting of the component items within the `<Composition>` list is determined by `Class` attribute, and is defined for each sub-class (e.g. See `<Composition Class = "Elemental"`).

Required attributes:

The `Unit` attribute

The units of the composition measurement *shall* be specified using the `Unit` attribute and *should* be "atoms", "mol%", "wt%", "vol%", or equivalent ppm/ppb units. The concentrations of all components (e.g. elements, compounds, isotopes, etc.) are uniformly defined using the same unit. Separate `Unit` attributes for each individual component *shall not* be used.

Example:

An example of the `<Composition>` condition is given for the `<Composition Class="Elemental">` sub-class.

<Composition Class="Elemental">

The `<Composition Class = "Elemental">` conditions template defines the composition of a material in terms of its constituent chemical elements.

Base template:

- `<Composition>`

Required elements:

The `<Element>` XML element

The `<Composition>` list *shall* contain one or more `<Element>` XML elements, like so:

```
<Element Z="11" DataType="float">3.</Element>
```

The atomic number of the chemical element defined by the `<Element>` XML element *shall* be specified by the `Z` attribute, and the numerical value of the `<Element>` XML element *shall* be the concentration of the chemical element, expressed in terms of the units specified by the `<Composition>` XML element.

Example:

```
<Composition Class="Elemental" Unit="wt%">
   <Element Z="19" DataType="float">14.05</Element>
   <Element Z="13" DataType="float">9.69</Element>
   <Element Z="14" DataType="float">30.27</Element>
   <Element Z="8" DataType="float">45.99</Element>
</Composition>
```

<DateTime>

The <DateTime> condition is a generic object to record the date and time of an event, such as the start time of an acquisition.

Optional elements:

The <Time>, <Date>, and <Timezone> elements

The <Date>, <Time> and <Timezone> elements are defined identically to the namesake elements in the <Header> list element (See Section 3.4: <Date>, <Time> and <Timezone>).

All three <Date>, <Time> and <Timezone> elements *should* be declared. A notable exception is when a time sequence of <DateTime> elements occur in a list, such as the <DateTimeList> element in the <Acquisition Class="IrregularArray"> condition template. In these instances, the <Date> and <Timezone> elements *should* be declared in the first <DateTime> element in the <DateTimeList>. Thereafter, the <Date> and <Timezone> elements may be omitted from subsequent <DateTime> entries if their values are unchanged from the preceding entry.

Example:

```
<DateTime>
  <Date>1985-10-26</Date>
  <Time>20:04:00</Time>
  <Timezone>UTC+10</Timezone>
</DateTime>
```

<Detector>

The `<Detector>` condition template is a generic object that describes the type and configuration of a detector used to collect a HMSA dataset. This template *should not* be used directly. Instead, use a sub-class appropriate for the type of detector, such as `<Detector Class="Spectrometer">`.

Optional elements:

The <Manufacturer> and <Model> elements

The name of the manufacturer of the detector and the model *should* be provided using the `<Manufacturer>` and `<Model>` elements, like so:

```
<Manufacturer>Example Inc.</Manufacturer>
<Model>Example Model 123</Model>
```

The <SerialNumber> element

The serial number of the detector may be provided as a text string using the `<SerialNumber>` element, like so:

```
<SerialNumber>12345-abc-67890</SerialNumber>
```

The `<SignalType>` element

The type of signal measured by the detector *should* be defined using the `<SignalType>` element, like so:

```
<SignalType>EDS</SignalType>
```

The recognised values of the `<SignalType>` element are the same as those defined in the EMSA/MAS spectrum file specification, namely:

- EDS = Energy Dispersive Spectroscopy
- WDS = Wavelength Dispersive Spectroscopy
- ELS = Energy Loss Spectroscopy
- AES = Auger Electron Spectroscopy
- PES = Photo Electron Spectroscopy
- XRF = X-ray Fluorescence Spectroscopy
- CLS = Cathodoluminescence Spectroscopy
- GAM = Gamma Ray Spectroscopy

Furthermore, this specification defines additional values for `<SignalType>` not included in the EMSA/MAS spectrum file specification:

- BEI = Backscattered Electron Imaging
- CBED = Convergent Beam Electron Diffraction
- EBSD = Electron Back-Scatter Diffraction
- EDIF = Electron Diffraction (generic)
- LEED = Low-Energy Electron Diffraction
- OPR = Optical, Reflected light
- OPT = Optical, Transmitted light
- PIXE = Proton Induced X-ray Emission Spectrometry
- RHEED = Reflection High-Energy Electron Diffraction
- SEI = Secondary Electron Imaging
- SXES = Soft X-ray Emission Spectroscopy

To propose additional values, please contact the HMSA working group.

The <MeasurementUnit> element

The physical units of the quantity measured by the detector (typically intensity) *should* be defined using the <MeasurementUnit>, like so:

```
<MeasurementUnit>counts</MeasurementUnit>
```

The value of the <MeasurementUnit> element *shall* be an SI, SI-derived or non-SI unit defined in Appendix C, or a compound unit thereof (See Section 2.5.4: Physical units). If the <MeasurementUnit> element is not specified, a default value of "counts" *shall* be assumed.

Note that for spectrometers, the units of the energy/wavelength calibration is separately defined using the <Calibration> element (see the <Detector Class="Spectrometer"> condition template in Appendix B).

The <Distance> element

The <Distance> element may be used to define the linear distance from the centre of the detector face to the point of incidence of the incident probe on the specimen surface. If provided, the <Distance> element *shall* be defined like so:

```
<Distance Unit="mm" DataType = "float">50</Distance>
```

The <Area> element

The <Area> element may be used to define the surface area of the detector element. If provided, the <Area> element *shall* be defined like so:

```
<Area Unit="mm2" DataType = "float">20</Area>
```

The <SolidAngle> element

The <SolidAngle> element may be used to define the solid angle subtended by the detector as measured from the point of incidence of the incident probe on the specimen surface. If provided, the <SolidAngle> element *shall* be defined like so:

```
<SolidAngle Unit="sr" DataType = "float">1.</SolidAngle>
```

The <SemiAngle> element

For radially-symmetric detectors facing at normal angles towards the specimen, the <SemiAngle> element may be used to define half of the inside cone angle subtended by the detector as measured from the point of incidence of the incident probe on the specimen surface. If provided, the <SemiAngle> element *shall* be defined like so:

```
<SemiAngle Unit="mrad" DataType = "float">3.4</SemiAngle>
```

The <Temperature> element

The <Temperature> element may be used to define the temperature of the detector element, like so:

```
<Temperature Unit="degreesC" DataType="float">-60.0</Temperature>
```

The <Elevation> element

The <Elevation> element *should* be used to define the elevation angle of the detector above the plane normal to the incident probe, as measured from the point of incidence on the specimen. As the <Elevation> element is measured relative to the incident probe, it is invariant with regards to rotation or tilt of the specimen. The <Elevation> element *shall* be measured in units of degrees, like so:

```
<Elevation Unit="degrees" DataType = "float">45.</Elevation>
```

The <Azimuth> element

The azimuthal angle of the detector may be defined using the <Azimuth> element, which is defined as the rotation around the incident probe vector, independent of specimen rotation or tilt. Zero azimuth occurs in the direction of the negative Y axis of the instrument (conventionally towards the rear of the instrument), and azimuth increases clockwise as viewed along the incident probe vector.

```
<Azimuth Unit="degrees" DataType = "float">0.</Azimuth>
```

The <DetectorName> element

The <DetectorName> element may be used to identify a particular detector on an instrument, such as distinguishing between the multiple WDS and/or EDS detectors on an EPMA. If provided, the <DetectorName> shall be defined as a text string like so:

```
<DetectorName>Ch1</DetectorName>
```

The value or interpretation of the <DetectorName> element are instrument specific, and are not defined in this specification.

Example:

Examples of the <Detector> condition template are given for the following sub-classes:

- <Detector Class="Camera">
- <Detector Class="Spectrometer/CL">
- <Detector Class="Spectrometer/WDS">
- <Detector Class="Spectrometer/XEDS">

<Detector Class="Camera">

The `<Detector Class="Camera">` condition template describes the calibration and collection mode of a camera used to collect a HMSA dataset, such as an EBSD or TEM camera.

Base template:

- `<Detector>`

Optional elements:

The `<FocalLength>` element

The focal length of the camera may be defined using the `<FocalLength>` element, like so:

```
<FocalLength Unit="mm" DataType="float">10.</FocalLength>
```

The `<ExposureTime>` element

The total exposure time of a frame, including integration, may be defined using the `<ExposureTime>` element, like so:

```
<ExposureTime Unit="ms" DataType="float">200.</ExposureTime>
```

The `<FrameIntegration>` element

The number of frames that are integrated to produce the output image may be defined using the `<FrameIntegration>` element, like so:

```
<FrameIntegration DataType="int">4</FrameIntegration>
```

The `<Magnification>` element

The nominal indicated magnification of the camera and optics may be defined using the `<Magnification>` element, like so:

```
<Magnification DataType="float">200.</Magnification>
```

The <NumericalAperture> element

The numerical aperture of the camera and optics may be defined using the <NumericalAperture> element, like so:

```
<NumericalAperture DataType="float">0.3</NumericalAperture>
```

Example:

Below is an example of a <Detector Class="Camera"> condition for a reflected-light optical microscope:

```
<Detector Class="Camera">
   <Manufacturer>Acme Optical Microscopes Pty. Ltd.</Manufacturer>
   <Model>9000</Model>
   <SerialNumber>0774-X11</SerialNumber>
   <SignalType>OPR</SignalType>
   <Elevation Unit="degrees" DataType = "float">90.</Elevation>
   <FocalLength Unit="mm" DataType="float">10.</FocalLength>
   <ExposureTime Unit="ms" DataType="float">40.</ExposureTime>
   <FrameIntegration DataType="int">1</FrameIntegration>
   <Magnification DataType="float">200.</Magnification>
   <NumericalAperture DataType="float">0.3</NumericalAperture>
</Detector>
```

<Detector Class="Spectrometer">

The `<Detector Class="Spectrometer">` condition template describes the calibration and collection mode of a spectrometer used to collect a HMSA dataset.

Note that the `<Detector Class="Spectrometer">` condition does not define the number of channels of the spectrometer. The number of channels is instead recorded in the datum dimensions of the associated spectral datasets.

Base template:

- `<Detector>`

Required elements:

The `<Calibration>` element

The calibration of the detector with respect to energy, wavelength, or other physical entities *shall* be defined using a `<Calibration>` element or a sub-class thereof (See the `<Calibration>` condition template). The `<Calibration>` element will be of the general form:

```
<Calibration Class="...">
   [...]
</Calibration>
```

If the spectrometer is operating as a monochromator (e.g. WDS mapping), the calibration definition *shall* be of the type `<Calibration Class="Constant">`.

Optional elements:

The <CollectionMode> element

The <CollectionMode> element may be provided to describe the time sequencing of the channel measurements of the spectrometer. If provided, the <CollectionMode> element shall be of the form:

```
<CollectionMode>Parallel</CollectionMode>
```

Recognised <CollectionMode> values include:

- "Parallel", for spectrometers that measure their entire range essentially simultaneously (e.g. XEDS, CCD), and;
- "Serial" for spectrometers that measure their range sequentially (e.g. WDS, AES, CL monochromators).

Example:

Examples of the <Detector Class="Spectrometer"> condition template are given for the following sub-classes:

- <Detector Class="Spectrometer/CL">
- <Detector Class="Spectrometer/WDS">
- <Detector Class="Spectrometer/XEDS">

<Detector Class="Spectrometer/CL">

The `<Detector Class="CL">` condition template describes the type and configuration of a cathodoluminescence spectrometer.

Base templates:

- `<Detector Class="Spectrometer">`
- `<Detector>`

Optional elements:

The `<DispersionElement>` element

The identity of the dispersion element of the CL spectrometer *should* be defined using the `<DispersionElement>` element, like so:

```
<DispersionElement>6M</DispersionElement>
```

This specification does not define recognised values for the `<DispersionElement>` element, as these are likely to be vendor-specific.

The `<Grating-d>` element

For CL spectrometers using diffraction gratings with constant d-spacing, the d-spacing *should* be defined using the `<Grating-d>` element, like so:

```
<Grating-d Unit="mm-1" DataType="float">800</Grating-d>
```

The d-spacing may be expressed as the distance between successive lines (`Unit`="um" or "nm"), or as lines per millimetre (`Unit`="mm-1").

The `<EntranceSlit>` element

The entrance slit of the spectrometer may be defined using the `<EntranceSlit>` element, like so:

```
<EntranceSlit Unit="um" DataType="float">50</EntranceSlit>
```

Example:

The example below is the definition of a Peltier-cooled grating CCD cathodoluminescence spectrometer, with a wavelength calibration defined as a 3rd order polynomial:

```
<Detector Class="Spectrometer/CL">
   <SignalType>CLS</SignalType>
   <CollectionMode>Parallel</CollectionMode>
   <Manufacturer>Acme Optical Instruments, GmbH</Manufacturer>
   <Model>MX800</Model>
   <DispersionElement>H50</DispersionElement>
   <Calibration Class="Polynomial">
      <Quantity>Wavelength</Quantity>
      <Unit>nm</Unit>
      <Coefficients DataType="array:float" Count="3">
         199.945602, 0.79385, -0.00003
      </Coefficients>
   </Calibration>
   <Temperature DataType="float" Unit="degreesC">-10</Temperature>
   <EntranceSlit DataType="float" Unit="um">200.</EntranceSlit>
</Detector>
```

<Detector Class="Spectrometer/WDS">

The <Detector Class="Spectrometer/WDS"> condition template describes the type and configuration of a wavelength dispersive x-ray spectrometer.

Base templates:

- <Detector Class="Spectrometer">
- <Detector>

Restrictions:

For WDS x-ray spectrometry, the <Calibration> object inherited from the <Detector Class="Spectrometer"> base class may be defined with <Quantity> values of:

- "Energy" (unit="eV"),
- "Wavelength" (unit="um", "nm", "Å", or "pm"),
- "2theta" (unit="degrees"), or;
- "Position" (unit="mm").

If the WDS spectrometer is performing a pulse height analyser (PHA) scan, the <Quantity> value may be:

- "PHA_Bias" (unit="V"),
- "PHA_Gain" (no units),
- "PHA_BaseLevel" (unit="V"), or;
- "PHA_Window" (unit="V").

Optional elements:

The `<DispersionElement>` element

The identity of the dispersion element *should* be defined using the `<DispersionElement>` element, like so:

```
<DispersionElement>TAP</DispersionElement>
```

The values of the `<DispersionElement>` may include traditional WDS crystals, such as:

- TAP, for thallium acid phthalate,
- RAP, for rubidium acid phthalate,
- KAP, for potassium acid phthalate,
- PET, for pentaerythritol, or;
- LIF, for lithium fluoride.

Additionally, the `<DispersionElement>` element may take values that are vendor-specific, including those of synthetic multi-layer elements (e.g. PC1, LDEB), or modified crystals (e.g. PETJ, LIFH).

The `<Crystal-2d>` element

The d-spacing of the dispersion element *should* be defined using the `<Crystal-2d>` element, like so:

```
<Crystal-2d Unit="Å" DataType="float">8.742</Crystal-2d>
```

The `<RowlandCircleDiameter>` element

The diameter of the Rowland circle of the WDS may be defined using the `<RowlandCircleDiameter>` element, like so:

```
<RowlandCircleDiameter Unit="mm" DataType="float">
   140.
</RowlandCircleDiameter>
```

The <PulseHeightAnalyser> elements

The configuration of the pulse height analyser (PHA) of the WDS spectrometer *should* be defined, like so:

```
<PulseHeightAnalyser>
   <Bias Unit="V" DataType="float">1700.</Bias>
   <Gain DataType="float">16.</Gain>
   <BaseLevel Unit="V" DataType="float">0.7</BaseLevel>
   <Window Unit="V" DataType="float">9.3</Window>
   <Mode>Integral</Mode>
</PulseHeightAnalyser>
```

The PHA <Mode> element *should* be either "Integral" or "Differential".

Note that if the WDS spectrometer is performing a PHA scan of the bias, gain, window or baseline, the variable parameter *shall* be omitted from the <PulseHeightAnalyser> definition. Instead, the variable parameter *shall* be defined as the calibration quantity in the <Calibration> object, as inherited from the base <Detector Class="Spectrometer"> condition template.

The <Counter> element

The configuration of the x-ray counter may be defined using the <Counter> element, like so:

```
<Counter>
   <Type>Sealed Xe</Type>
   <Pressure Unit="Pa" DataType="float">1.01E5</Pressure>
   <Window>
      <Material>
         <MaterialName>Al</MaterialName>
         <Thickness Unit="nm" DataType="float">40</Thickness>
      </Material>
      <Material>
         <MaterialName>PET</MaterialName>
         <Thickness Unit="nm" DataType="float">200</Thickness>
      </Material>
      [ multiple layers are supported ]
   </Window>
</Counter>
```

The counter <Type> element for proportional counters may be "Sealed" or "Flow", followed by the gas medium, which may be "Xe", "Ar", and "P10" (for 10% methane in Ar). Scintillator-photomultiplier counters are supported, but are not defined here.

The <Pressure> element may be used to define the gas pressure in a proportional counter. If the counter pressure value is regulated relative to ambient atmospheric pressure, such as may occur with a flow type counter, the <Pressure> element *shall* include a Relative="Ambient" attribute. If this attribute is not present, the pressure *shall* be interpreted as absolute.

The <Window> and <Material> elements are defined identically to those in the <Detector Class="Spectrometer/XEDS"> condition template, including the list of recognised window materials (e.g. "Al", "Be", "PET", etc.)

The <WDSPosition> element

When the WDS spectrometer is performing a PHA scan of the bias, gain, window or baseline, the PHA scan parameter is defined as the calibration quantity in the <Calibration> object, as inherited from the base <Detector Class="Spectrometer"> condition template. Therefore, the WDS dispersion element position or angle *shall* be separately defined using the <WDSPosition> element, as below:

```
<WDSPosition Unit="mm" DataType="float">95</WDSPosition>
```

The Unit attribute may be "mm" for spectrometers with linear positions, or "degrees" for spectrometers with angular (2 theta) positions. Note that the <WDSPosition> element should not be confused with the <Position> element (defined in the base <Detector> condition template), which defines the linear distance from the specimen to the detector.

Examples:

The example below is a WDS detector functioning as a monochromator, such as when performing WDS elemental mapping.

```
<Detector Class="Spectrometer/WDS">
   <DetectorName>Ch1</DetectorName>
   <SignalType>WDS</SignalType>
   <DispersionElement>LIF</DispersionElement>
   <Crystal-2d DataType="float" Unit="Å">4.0267</Crystal-2d>
   <Calibration Class="Constant">
      <Quantity>Position</Quantity>
      <Unit>mm</Unit>
      <Value DataType="float">178.255997</Value>
      </Calibration>
   <Elevation DataType="float" Unit="degrees">40.</Elevation>
   <RowlandCircleDiameter DataType="float" Unit="mm">
      140.
   </RowlandCircleDiameter>
   <PulseHeightAnalyser>
      <Bias DataType="float" Unit="V">1623.</Bias>
      <Gain DataType="float">16.</Gain>
      <BaseLevel DataType="float" Unit="V">2.</BaseLevel>
      <Window DataType="float" Unit="V">4.</Window>
      <Mode>Differential</Mode>
   </PulseHeightAnalyser>
   <Counter>
      <Type>Flow P10</Type>
   </Counter>
</Detector>
```

The example below is a WDS detector measuring peak and backgrounds, such as when performing WDS quantitative analysis. Position units are defined in terms of mm, and the position of the peak, low background and high background are explicitly defined in the `<Calibration>` element.

```
<Detector Class="Spectrometer/WDS">
   <DetectorName>Ch2</DetectorName>
   <SignalType>WDS</SignalType>
   <DispersionElement>TAP</DispersionElement>
   <Crystal-2d DataType="float" Unit="Å">25.75</Crystal-2d>
   <Calibration Class="Explicit">
      <Quantity>Position</Quantity>
      <Unit>mm</Unit>
      <Values DataType="array:float" Count="3">
         77.648003, 76.648003, 78.648003
      </Values>
      <Labels>Peak, BgndLow, BgndHigh</Labels>
   </Calibration>
   <Elevation DataType="float" Unit="degrees">40.</Elevation>
   <RowlandCircleDiameter DataType="float" Unit="mm">
      140.
   </RowlandCircleDiameter>
   <PulseHeightAnalyser>
      <Bias DataType="float" Unit="V">1700.</Bias>
      <Gain DataType="float">32.</Gain>
      <BaseLevel DataType="float" Unit="V">1.5</BaseLevel>
      <Window DataType="float" Unit="V">3.5</Window>
      <Mode>Differential</Mode>
   </PulseHeightAnalyser>
   <Counter>
      <Type>Flow P10</Type>
   </Counter>
</Detector>
```

The example below is a WDS detector performing a wavescan over a range of 2 theta angles, starting at 45 degrees and stepping in increments of 0.01 degrees.

```
<Detector Class="Spectrometer/WDS">
   <DetectorName>Ch2</DetectorName>
   <SignalType>WDS</SignalType>
   <DispersionElement>TAP</DispersionElement>
   <Crystal-2d DataType="float" Unit="Å">25.75</Crystal-2d>
   <Calibration Class="Linear">
      <Quantity>2theta</Quantity>
      <Unit>degrees</Unit>
      <Offset DataType="float">45</Offset>
      <Gain DataType="float">0.01</Gain>
   </Calibration>
   <Elevation DataType="float" Unit="degrees">40.</Elevation>
   <RowlandCircleDiameter DataType="float" Unit="mm">
      140.
   </RowlandCircleDiameter>
   <PulseHeightAnalyser>
      <Bias DataType="float" Unit="V">1700.</Bias>
      <Gain DataType="float">32.</Gain>
      <BaseLevel DataType="float" Unit="V">1.5</BaseLevel>
      <Window DataType="float" Unit="V">3.5</Window>
      <Mode>Differential</Mode>
   </PulseHeightAnalyser>
   <Counter>
      <Type>Sealed Xe</Type>
   </Counter>
</Detector>
```

The example below is a WDS detector performing a PHA window scan starting at 1V, in steps of 50mV. Note that the <Calibration> object is now defined in terms of PHA window volts, the analyser position is now defined using the <WDSPosition> element (here as degrees 2-theta), and the <Window> is not specified in the <PulseHeightAnalyser> element. The dataset associated with the below <Detector Class="Spectrometer/WDS"> condition would then store the detected counts at each 50mV step, in the same way that an EELS or XEDS spectrum would be stored.

```
<Detector Class="Spectrometer/WDS">
   <DetectorName>Ch2</DetectorName>
   <SignalType>WDS</SignalType>
   <DispersionElement>TAP</DispersionElement>
   <Crystal-2d DataType="float" Unit="Å">25.75</Crystal-2d>
   <WDSPosition DataType="float" Unit="degrees">45.0</WDSPosition>
   <Calibration Class="Linear">
      <Quantity>PHA_Window</Quantity>
      <Unit>V</Unit>
      <Offset DataType="float">1</Offset>
      <Gain DataType="float">0.05</Gain>
   </Calibration>
   <Elevation DataType="float" Unit="degrees">40.</Elevation>
   <RowlandCircleDiameter DataType="float"
Unit="mm">140.</RowlandCircleDiameter>
   <PulseHeightAnalyser>
      <Bias DataType="float" Unit="V">1700.</Bias>
      <Gain DataType="float">32.</Gain>
      <BaseLevel DataType="float" Unit="V">1.5</BaseLevel>
      <Mode>Differential</Mode>
   </PulseHeightAnalyser>
   <Counter>
      <Type>Flow P10</Type>
   </Counter>
</Detector>
```

<Detector Class="Spectrometer/XEDS">

The `<Detector Class="Spectrometer/XEDS">` condition template describes the type and configuration of an energy dispersive x-ray spectrometer.

Base templates:

- `<Detector Class="Spectrometer">`
- `<Detector>`

Optional elements:

The `<Technology>` element

The XEDS detector technology type may be defined using the `<Technology>` element, like so:

```
<Technology>SDD</Technology>
```

Recognised values of the `<Technology>` element include:

- "Ge", for germanium based detectors
- "SiLi", for lithium-drifted silicon detectors
- "SDD", for silicon drift detectors
- "Microcalorimeter"

The `<NominalThroughput>` element

The nominal maximum throughput of the XEDS spectrometer, in terms of kilocounts per second, may be defined using the `<NominalThroughput>` element like so:

```
<NominalThroughput Unit="kcounts/s" DataType="float">
   180.
</NominalThroughput>
```

The `<TimeConstant>` element

The time constant of the XEDS detector preamplifier may be defined using the `<TimeConstant>` element like so:

```
<TimeConstant Unit="us" DataType="float">12.5</TimeConstant>
```

The `<StrobeRate>` element

The nominal input count rate of the zero-energy noise strobe of the detector, if known, *should* be defined using the `<StrobeRate>` element like so:

```
<StrobeRate Unit="Hz" DataType="float">2000</StrobeRate>
```

The <Window> element

The composition and thickness of the window in front of the XEDS detector may be defined using the <Window> element, which contains one or more <Material> elements (see <Material> condition template), likes so:

```
<Window>
   <Material>
      <MaterialName>Al</MaterialName>
      <Thickness Unit="nm" DataType="float">40</Thickness>
   </Material>
   <Material>
      <MaterialName>PET</MaterialName>
      <Thickness Unit="nm" DataType="float">200</Thickness>
   </Material>
   [ multiple layers are supported ]
</Window>
```

The order of the window layer materials defined in the <Window> list *shall* be given on the order from the specimen side to the detector side.

The composition of the window materials may be defined exhaustively using the <Composition> optional element in the <Material> condition template, or by providing the material name using the <MaterialName> element. Recognised window materials are:

- "Al", for aluminium windows
- "Be", for beryllium windows
- "BN", for boron nitride windows
- "Diamond", for diamond windows
- "PET", for polyethylene terephthalate windows
- "PP", for polypropylene windows

Vendor-specific values of <MaterialName> may be used when customary to do so.

The <GoldLayer> element

The thickness of the gold layer / electrical contact on the surface of the XEDS detector may be defined using the <GoldLayer> element like so:

```
<GoldLayer Unit="nm" DataType="float">20.0</GoldLayer>
```

The <DeadLayer> element

The thickness of the dead layer of the XEDS detector may be defined using the <DeadLayer> element like so:

```
<DeadLayer Unit="nm" DataType="float">100.0</DeadLayer>
```

The <ActiveLayer> element

The thickness of the active layer of the XEDS detector may be defined using the <ActiveLayer> element like so:

```
<ActiveLayer Unit="mm" DataType="float">3.0</ActiveLayer>
```

Examples:

The following example is a typical SDD detector:

```
<Detector Class="Spectrometer/XEDS">
   <Manufacturer>Acme Detector Co.</Manufacturer>
   <Model>Acme Model 1</Model>
   <SignalType>EDS</SignalType>
   <MeasurementUnit>counts</MeasurementUnit>
   <CollectionMode>Parallel</CollectionMode>
   <Technology>SDD</Technology>
   <Calibration Class="Linear">
      <Quantity>Energy</Quantity>
      <Unit>eV</Unit>
      <Gain DataType="float">10.</Gain>
      <Offset DataType="float">-475.</Offset>
   </Calibration>
   <Temperature Unit="degreesC" DataType="float">-25.0</Temperature>
   <NominalThroughput DataType="float" Unit="kcounts/s">
      400.
   </NominalThroughput>
   <StrobeRate DataType="float" Unit="Hz">2000.</StrobeRate>
   <Area DataType="float" Unit="mm2">10.</Area>
   <Elevation DataType="float" Unit="degrees">45.</Elevation>
   <Window>
       <Material>
         <MaterialName>Al</MaterialName>
         <Thickness Unit="nm" DataType="float">40</Thickness>
      </Material>
      <Material>
         <MaterialName>PET</MaterialName>
         <Thickness Unit="nm" DataType="float">200</Thickness>
      </Material>
   </Window>
</Detector>
```

<ElementalID>

The `<ElementalID>` condition template defines an elemental identification, as may be useful for region of interest images, WDS elemental maps, XAFS spectral maps, etc.

Required elements:

The `<Element>` element

The `<Element>` element defines the chemical element of the `<ElementalID>` object, using the chemical symbol as a text string, like so:

```
<Element>Na</Element>
```

Example:

An example of the `<ElementalID>` condition, as applied to an x-ray line identification, is given for the `<ElementalID Class="X-ray">` sub-class.

<ElementalID Class="X-ray">

The `<ElementalID Class="X-ray">` condition template defines and elemental identification based on an x-ray peak, as may be useful for region of interest images and the like.

Base template:

- `<ElementalID>`

Required elements:

The `<Line>` element

The name of the x-ray line *shall* be provided using the `<Line>` element, like so:

```
<Line>Ma</Line>
```

X-ray line names may be given in Siegbahn or IUPAC naming conventions. The notation may be specified using the `Notation` attribute, and alternative notations may be provided using the `alt-IUPAC` and `alt-Siegbahn` attributes, as shown below.

```
<Line Notation = "IUPAC" alt-Siegbahn = "Ma">M5-N6,7</Line>
<Line Notation = "Siegbahn" alt-IUPAC = "L2-M4">Lb1</Line>
```

For compatibility reasons, the Siegbahn notation *shall* use the Latin characters a, b, g, z and n in place of the Greek α, β γ, ζ and η. Similarly, for the IUPAC notation, numerals *shall not* be given in subscripts.

A list of principal x-ray lines in both IUPAC and Siegbahn notations is given below. For a complete list of corresponding transition names between IUPAC and Siegbahn notations, please refer to:

- Jenkins R., Manne R., Robin R. and Senemaud C., *Nomenclature System for X-Ray Spectroscopy*, X-RAY SPECTROM., VOL. 20, pp149-155 (1991).

K-series		**L-series**		**M-series**	
Siegbahn	**IUPAC**	**Siegbahn**	**IUPAC**	**Siegbahn**	**IUPAC**
Ka	K-L2,3	La	L3-M4,5	Ma	M5-N6,7
Ka1	K-L3	La1	L3-M5	Ma1	M5-N7
Ka2	K-L2	La2	L3-M4	Ma2	M5-N6
Kb	K-M,N	Lb1	L2-M4	Mb	M4-N6
Kb1	K-M3	Lb2	L3-N5	Mg	M3-N5
Kb2	K-N2,3	Lb3	L1-M3	Mz	M4,5-N2,3
		Lb4	L1-M2		
		Lg1	L2-N4		
		Lg2	L1-N2		
		Lg3	L1-N3		
		Ll	L3-M1		
		Ln	L2-M1		

Optional elements:

The `<Energy>` element

The energy of the x-ray line may be specified using the `<Energy>` element, like so:

```
<Energy Unit="eV" DataType="float">1234</Energy>
```

Example:

```
<ElementalID Class="X-ray">
   <Element>Al</Element>
   <Line>La</Line>
   <Energy Unit="eV" DataType="float">73</Energy>
</ElementalID>
```

<Instrument>

The `<Instrument>` condition template is a generic object that describes the type of instrument used to collect a HMSA dataset.

Required elements:

The `<Manufacturer>` and `<Model>` elements

The name of the manufacturer of the instrument and the model *shall* be provided using the `<Manufacturer>` and `<Model>` elements, like so:

```
<Manufacturer>Example Inc.</Manufacturer>
<Model>Example Model 123</Model>
```

Optional elements:

The `<SerialNumber>` element

The serial number of the instrument may be provided as a text string using the `<SerialNumber>` element, like so:

```
<SerialNumber>12345-abc-67890</SerialNumber>
```

Example:

```
<Instrument>
   <Manufacturer>Acme Microscopes Inc.</Manufacturer>
   <Model>Model 6000</Model>
   <SerialNumber>12345-abc-67890</SerialNumber>
</Instrument>
```

<Material>

The <Material> condition defines the composition of a material.

Optional elements:

The <MaterialName> element

The name of the material may be defined using the <MaterialName> element, like so:

```
<MaterialName>Orthoclase</MaterialName>
```

The <Formula> element

The chemical formula of the material may be defined using the <Formula> element, like so:

```
<Formula>KAlSi3O8</Formula>
```

The <Composition> element

The <Composition> list may be used to store the abundance of each component of the composition, using a <Composition> condition object, like so:

```
<Composition Class = "Elemental" Unit="wt%">
   [...]
</Composition>
```

The <Density> element

The density of the material may be defined using the <Density> element, like so:

```
<Density Unit="g/cm3" DataType="float">2.56</Density>
```

The value of the Units attribute may be the SI unit of kilograms per cubic metre ("kg/m3"), or the customary SI-related metric units of grams per cubic centimetre ("g/cm3").

The `<Thickness>` element

If the thickness of the material is known, it may be defined using the `<Thickness>` element, like so:

```
<Thickness Unit="um" DataType="float">2.5</Thickness>
```

Example:

```
<Material>
   <MaterialName>Orthoclase</MaterialName>
   <Formula>KAlSi3O8</Formula>
   <Composition Class="Elemental" Unit="wt%">
      <Element Z="19" DataType="float">14.05</Element>
      <Element Z="13" DataType="float">9.69</Element>
      <Element Z="14" DataType="float">30.27</Element>
      <Element Z="8" DataType="float">45.99</Element>
   </Composition>
   <Density Unit="g/cm3" DataType="float">2.56</Density>
</Material>
```

<Probe>

The `<Probe>` condition template is a generic object that describes the type and conditions of the analytical probe used to collect a HMSA dataset, such as settings for electron or ion columns, lasers, etc. This template *should not* be used directly. Instead, use a sub-class appropriate for the type of probe, such as `<Probe Class="EM">`.

Optional elements:

The <ConvergenceAngle> element

The convergence semi-angle of the incident probe at the specimen may be defined using the `<ConvergenceAngle>` element, like so:

```
<ConvergenceAngle Unit="mrad" DataType="float">1.5</ConvergenceAngle>
```

The <WorkingDistance> element

The working distance from the last probe optic element to the specimen surface may be defined using the `<WorkingDistance>` element, like so:

```
<WorkingDistance DataType="float" Unit="mm">10</WorkingDistance>
```

<Probe Class="EM">

The `<Probe Class="EM">` condition template describes the electron column conditions of the electron microscope used to collect a HMSA dataset.

Base template:

- `<Probe>`

Required elements:

The `<BeamVoltage>` element

The accelerating voltage of the electron column, measured as landing voltage at the specimen, *shall* be defined using the `<BeamVoltage>` element like so:

```
<BeamVoltage DataType="float" Unit="kV">15.</BeamVoltage>
```

Optional elements:

The `<BeamCurrent>` element

The incident probe current, if known, *should* be defined using the `<BeamCurrent>` element like so:

```
<BeamCurrent DataType="float" Unit="nA">47.59</BeamCurrent>
```

The <GunType> element

The type of electron gun *should* be defined using the <GunType> element like so:

```
<GunType>Schottky FEG</GunType>
```

Recognised values of the gun type are:

- "W filament", for conventional tungsten filament thermionic emitters,
- "LaB6", for lanthanum hexaboride thermionic emitters,
- "Cold FEG", for cold-cathode field emission guns, and;
- "Schottky FEG", for warm field emission guns.

The <EmissionCurrent> element

The emission current of the electron gun may be defined using the <EmissionCurrent> element like so:

```
<EmissionCurrent DataType="float" Unit="uA">12345</EmissionCurrent>
```

The <FilamentCurrent> element

The emitter filament heater current may be defined using the <FilamentCurrent> element like so:

```
<FilamentCurrent DataType="float" Unit="A">1.234</FilamentCurrent>
```

The <ExtractorBias> element

The bias voltage on the gun extractor may be defined using the <ExtractorBias> element like so:

```
<ExtractorBias DataType="float" Unit="V">4200</ExtractorBias>
```

The <BeamDiameter> element

The diameter of the electron beam at the point of incidence on the specimen may be defined using the <BeamDiameter> element like so:

```
<BeamDiameter DataType="float" Unit="nm">12345</BeamDiameter>
```

The <ChamberPressure> element

The specimen chamber vacuum pressure may be defined using the <ChamberPressure> element like so:

```
<ChamberPressure DataType="float" Unit="Pa">3.14E-6</ChamberPressure>
```

The <GunPressure> element

The gun chamber vacuum pressure may be defined using the <GunPressure> element like so:

```
<GunPressure DataType="float" Unit="Pa">3.14E-10</GunPressure>
```

The <ScanMagnification> element

The nominal scan magnification of a scanning electron column (e.g. SEM, STEM) may be defined using the <ScanMagnification> element like so:

```
<ScanMagnification DataType="float">2500.</ScanMagnification>
```

The <CondenserApertureDiameter> element

The diameter of the condenser lens aperture may be defined using the <CondenserApertureDiameter> element like so:

```
<CondenserApertureDiameter DataType="float" Unit="um">
   100
</CondenserApertureDiameter>
```

The <ObjectiveApertureDiameter> element

The diameter of the objective lens aperture may be defined using the <ObjectiveApertureDiameter> element like so:

```
<ObjectiveApertureDiameter DataType="float" Unit="um">
   100
</ObjectiveApertureDiameter>
```

Example:

Below is an example of the `<Probe Class="EM">` condition for a warm field emission electron column such as an SEM/EPMA:

```
<Probe Class="EM">
   <GunType>Schottky FEG</GunType>
   <BeamVoltage DataType="float" Unit="kV">15.</BeamVoltage>
   <BeamCurrent DataType="float" Unit="nA">47.59</BeamCurrent>
   <WorkingDistance DataType="float" Unit="mm">10</WorkingDistance>
   <ScanMagnification DataType="float">2500.</ScanMagnification>
   <ChamberPressure DataType="float" Unit="Pa">1.4E-6</ChamberPressure>
   <GunPressure DataType="float" Unit="Pa">2.7E-9</GunPressure>
   <BeamDiameter DataType="float" Unit="nm">10</BeamDiameter>
   <ExtractorBias DataType="float" Unit="V">4200</ExtractorBias>
   <FilamentCurrent DataType="float" Unit="A">2.4</FilamentCurrent>
   <EmissionCurrent DataType="float" Unit="uA">240</EmissionCurrent>
</Probe>
```

<Probe Class="EM/TEM">

The `<Probe Class="EM/TEM">` condition template extends on the `<Probe Class="EM">` template to define additional electron column conditions of transmission electron microscopes.

Note that camera-related conditions such as magnification *should* be stored in a `<Detector Class="Camera">` condition template.

Base templates:

- `<Probe Class = "EM">`
- `<Probe>`

Required elements:

The <OperatingMode> element

The operating mode of the TEM *shall* be defined using the `<OperatingMode>` element, like so:

```
<OperatingMode>IMAGE</OperatingMode>
```

The recognised values of the `<OperatingMode>` element are the same as those defined in the EMSA/MAS spectrum file specification, namely:

- "IMAGE", for imaging mode,
- "DIFFR", for diffraction mode,
- "SCIMG", for scanning imaging mode, and;
- "SCDIF", for scanning diffraction mode.

Example:

Below is an example of the `<Probe Class="EM/TEM">` condition for a FEG TEM operating in STEM mode:

```
<Probe Class="EM">
   <GunType>Schottky FEG</GunType>
   <BeamVoltage DataType="float" Unit="kV">300.</BeamVoltage>
   <BeamCurrent DataType="float" Unit="pA">50</BeamCurrent>
   <OperatingMode>SCIMG</OperatingMode>
   <ConvergenceAngle Unit="mrad" DataType="float">1.5</ConvergenceAngle>
   <WorkingDistance DataType="float" Unit="mm">5</WorkingDistance>
   <ScanMagnification DataType="float">1250000.</ScanMagnification>
   <ChamberPressure DataType="float" Unit="Pa">1.4E-8</ChamberPressure>
   <GunPressure DataType="float" Unit="Pa">2.7E-9</GunPressure>
   <BeamDiameter DataType="float" Unit="nm">0.1</BeamDiameter>
   <ExtractorBias DataType="float" Unit="V">4200</ExtractorBias>
   <FilamentCurrent DataType="float" Unit="A">2.4</FilamentCurrent>
   <EmissionCurrent DataType="float" Unit="uA">5.5</EmissionCurrent>
</Probe>
```

<RegionOfInterest>

The <RegionOfInterest> condition template defines a region of a spectrum (or other one-dimensional datum), as may be useful for defining start and end channels used for a region of interest image.

Required elements:

The <StartChannel> and <EndChannel> elements

The start and end channel (inclusive) of a region of interest *shall* be defined using the <StartChannel> and <EndChannel> elements, like so:

```
<StartChannel DataType="int">556</StartChannel>
<EndChannel DataType="int">636</EndChannel>
```

The value of <StartChannel> must be equal to or greater than 0, and smaller than or equal to the value of <EndChannel>.

The calibration of the ROI channels with respect to energy, wavelength, etc, may be determined from the <Calibration> element defined in the <Detector Class="Spectrometer"> condition associated with the dataset.

Example:

Below is an example of a region of interest definition, such as may be used with an elemental x-ray map.

```
<RegionOfInterest Name="Fe Ka">
   <StartChannel DataType="int">556</StartChannel>
   <EndChannel DataType="int">636</EndChannel>
</RegionOfInterest>
```

Note that, notwithstanding the Name attribute in the example above, the element or line of an elemental map dataset should be determined from an <ElementalID Class="X-ray"> condition. The <RegionOfInterest> condition *only* defines the channel range of the spectrum used for the ROI image.

<Specimen>

The `<Specimen>` conditions template defines a physical specimen, including the name, origin, composition, etc.

Optional elements:

The `<Name>` element

The name or identifier of the specimen *should* be provided using the `<Name>` element, like so:

```
<Name>Cryolite EPMA standard #1</Name>
```

The `<Description>` element

A description of the specimen may be provided using the `<Description>` element, like so:

```
<Description>Natural cryolite standard</Description>
```

The `<Owner>` element

The owner of the specimen may be provided using the `<Owner>` element, like so:

```
<Owner>Acme Geosciences Laboratory plc</Owner>
```

The `<Origin>` element

The origin (i.e. geographical location, source institution / vendor, etc.) may provided using the `<Origin>` element, like so:

```
<Origin>Ilimaussaq complex, Narsaq peninsula, Greenland</Origin>
```

The <Material> element

If the specimen composition is known and uniform, the material may be defined using a <Material> element, like so:

```
<Material>
   <Formula>Na3AlF6</Formula>
   <Composition Class="Elemental" Unit="atoms">
      <Element Z="11" DataType="float">3</Element>
      <Element Z="13" DataType="float">1</Element>
      <Element Z="9" DataType="float">6</Element>
   </Composition>
   <Density Unit="g/cm3" DataType="float">2.97</Density>
</Material>
```

The <Coating> element

If one or more coatings were applied to the analysis surface of the specimen, the coating(s) *should* be defined using a <Coating> element containing one or more <Material> elements, like so:

```
<Coating>
   <Material>
      <MaterialName>Carbon</MaterialName>
      <Thickness Unit="nm" DataType="float">50</Thickness>
   </Material>
   [ Multiple coatings are allowed ]
</Coating>
```

If multiple coating layers are present, the order of <Material> elements in the <Coating> *shall* be the order of the layers from top to bottom (i.e. towards the substrate).

The <Thickness> element

If the specimen thickness (excluding coatings) is known and uniform, it may be defined using the <Thickness> element, like so:

```
<Thickness Unit="nm" DataType="float">10.0</Thickness>
```

The `<Temperature>` element

The temperature of the specimen during analysis may be provided using the `<Temperature>` element, like so:

```
<Temperature Unit="degreesC" DataType="float">-20.0</Temperature>
```

Example:

The following is an example of a bulk specimen of cryolite (Na_3AlF_6), which has been carbon coated, and cooled to liquid nitrogen temperature.

```
<Specimen>
   <Name>Cryolite EPMA standard #1</Name>
   <Description>Natural cryolite standard</Description>
   <Owner>Acme Geosciences Laboratory plc</Owner>
   <Origin>Ilimaussaq complex, Narsaq peninsula, Greenland</Origin>
   <Material>
      <Formula>Na3AlF6</Formula>
      <Density Unit="g/cm3" DataType="float">2.97</Density>
   </Material>
   <Coating>
      <Material>
         <MaterialName>Carbon</MaterialName>
         <Thickness Unit="nm" DataType="float">50</Thickness>
      </Material>
   </Coating>
   <Temperature Unit="K" DataType="float">77</Temperature>
</Specimen>
```

<SpecimenPosition>

The <SpecimenPosition> condition template defines a physical location on, or in, the specimen.

Coordinate system:

The default coordinate system for the <SpecimenPosition> element is defined in terms of the probe geometry of the instrument. The Z axis is defined by the probe direction, such that when the analysis surface of the specimen is placed at normal incidence to the incident probe, the positive Z axis is pointing into the specimen. The X and Y axes are orthogonal to the Z axis and to each other, and *shall* be oriented such that they form a 'right handed' coordinate system. For vertically oriented probes, with the positive probe axis pointing downwards, the positive Y axis *should* point towards the front of the instrument, and the positive X axis *should* point to the right of the instrument. The rotation of the X and Y axes around the Z axis relative to the instrument is not defined for other probe orientations, and may be assigned arbitrarily, providing 'right handedness' is preserved. This convention corresponds to the coordinate system used in some microscopes, and is consistent with the customary coordinate system used in computer graphics, where X is to the right, Y is down the screen, and Z is into the screen (also a 'right handed' system).

Note that the specimen coordinate system may be arbitrarily tilted and/or rotated from this default orientation relative to the probe using the <EulerRotation> (for tilt) and <R> (for rotation) elements as described below.

The order of the translation (<X>, <Y> and <Z>), tilt (<EulerRotation>), and Z-axis rotation (<R>) operations that map from default probe coordinates to specimen coordinates is determined by the order of the <X>, <Y> <Z>, <EulerRotation> and <R> elements as they are listed in the <SpecimenPosition> condition. See example below.

Optional elements:

The <X>, <Y> and <Z> elements

The coordinate in the X, Y and Z Cartesian axes may be defined using the <X>, <Y> and <Z> elements, like so:

```
<X Unit="mm" DataType="float">1.0</X>
<Y Unit="mm" DataType="float">-5.0</Y>
<Z Unit="mm" DataType="float">10.0</Z>
```

The origins of the X, Y and Z axes in the specimen coordinate system are arbitrary, and may vary from instrument to instrument depending on the implementation of the probe deflection and focussing system, and/or the specimen stage.

The <EulerRotation> element

To support a range of specimen tilting apparatus, such as double-tiling TEM holders, eucentric goniometer stages, as well as SEM tilting stages with different tilt axes, the tilt of the specimen may defined as a rotation by up to three Euler angles, using a <EulerRotation> element, like so:

```
<EulerRotation Sequence="z-x-z" Unit="degrees" DataType="array:float"
Count="3">
   30, 10, 0
</EulerRotation>
```

The Sequence attribute defines the extrinsic Euler rotation operation, and may include *up to* three of the x, y and z axes in any combination. In the above example using the "z-x-z" sequence, the first rotation is about the Z axis (clockwise from above), the second rotation is about the transformed X axis (clockwise from the left), and the third rotation is about the transformed Z axis (clockwise from above). In all cases, the rotations are ‘right-handed’. The number of angles in the <EulerRotation> element may be one, two or three, and *shall* be equal to the number of transformations in the Sequence attribute. For example, if two Euler angles are defined, the Sequence shall contain two transformations (e.g. "x-y").

Note that for single-axis rotations, the <EulerRotation> value is a single value, not an array.

The <R> element

The rotation coordinate around the normal to the specimen surface may be defined using the <R> element, like so:

```
<R Unit="degrees" DataType="float">90.0</R>
```

The rotation operation rotates the physical specimen around the Z axis of the specimen coordinate system, in a clockwise direction when viewed from above.

The origin of the <R> axis is not defined in this condition, and may be considered arbitrary.

Example:

The following is an example of a stage condition for a typical EBSD experiment in an SEM, where the rotation axis is mounted on top of the X/Y/Z translation stage, which is in turn mounted on top of the tilt axis. If the specimen is tilted 70 degrees towards the EBSD detector, which is mounted in the direction of the negative Y axis relative to the specimen (i.e. rear of instrument, by default), and the stage rotation is set to 15 degrees clockwise, the resulting <SpecimenPosition> condition will have the following order and form:

```
<SpecimenPosition>
   <EulerRotation Sequence="x" Unit="degrees" DataType="float">
      -70
   </EulerRotation>
   <X Unit="mm" DataType="float">1.0</X>
   <Y Unit="mm" DataType="float">-5.0</Y>
   <Z Unit="mm" DataType="float">20</Z>
   <R Unit="degrees" DataType="float">15.0</R>
</SpecimenPosition>
```

The sign of the 70 degree rotation is negative because a tilt towards the negative Y axis is counter-clockwise when viewed along the positive X direction (i.e. from left to right).

Appendix C - Units and prefixes

Parameters in HMSA XML files *shall* use SI units, SI derived units, or a limited set of non-SI units defined below. Except where noted below, all SI magnitude prefixes are permitted for all units (e.g. mm, keV).

Units and prefixes are case sensitive, and *shall* be written with appropriate capitalisation as given below.

SI units

Symbol	Unit	Quantity
m	Metre	Length
kg	Kilogram	Mass. Prefixes *may* be used for values smaller than one kilogram (e.g. mg, ng), but *shall not* for values larger than one kilogram (no Gg, etc.)
s	Second	Time. Prefixes *may* be used for values smaller than one second (e.g. ns, ms), but *shall not* for values larger than one second (no ks, Ms, etc.)
A	Ampere	Current
K	Kelvin	Temperature
mol	Mole	Amount of substance
Cd	Candela	Luminous intensity

Si-derived units

Symbol	Unit	Quantity
Å	Ångström	Length, equivalent to 10^{-10}m. Prefixes are not permitted (e.g. no kÅ). Note the character used is the Latin letter A with ring above (U+00C5). If 'Å' is untypeable, use appropriate conversions to 'nm' or 'pm', but never 'A'.
Bq	Becquerel	Radioactivity, equivalent to s^{-1}.
C	Coulomb	Electrical charge, equivalent to A.s.
Da	Dalton	Atomic mass, equivalent to $1.66053886 \times 10^{-27}$ kg.
degreesC	Degree Celsius	Temperature, equivalent to K + 273.15. For compatibility reasons, 'degreesC' *should* be used in place of the Unicode degree symbol (U+00B0), as in '°C'.
F	Farad	Capacitance, equivalent to C/V.
Gy	Gray	Absorbed dose, equivalent to J/kg.
H	Henry	Inductance, equivalent to Wb/A.
Hz	Hertz	Frequency, equivalent to s^{-1}.
J	Joule	Energy, equivalent to $kg.m^2/s^2$.
L	Litre	Volume, equivalent to 10^{-3} m^3.
lm	Lumen	Luminous flux, equivalent to W/m^2.
lx	Lux	Illuminance, equivalent to lm/m^2.
N	Newton	Force, equivalent to $kg.m/s^2$.
Ohm	Ohm	Electrical resistance, equivalent to V/A. For compatibility reasons, 'Ohm' *should* be used in place of the Unicode Greek capital letter omega 'Ω' (U+03A9).

Pa	Pascal	Pressure, equivalent to N/m^2.
rad	Radian	Angle.
S	Siemens	Electrical conductance, equivalent to Ohm^{-1}.
Sv	Sievert	Equivalent dose.
sr	Steradian	Solid angle.
T	Tesla	Magnetic flux density, equivalent to Wb/m^2.
V	Volt	Electrical potential, equivalent to J/C.
W	Watt	Power, equivalent to J/s.
Wb	Weber	Magnetic flux, equivalent to J/A.

Non-SI units

Symbol	Unit	Quantity
degrees	Degree	Angle. For compatibility reasons, ‘degrees’ *should* be used in place of the Unicode degree symbol ‘°’ (U+00B0).
atoms	Number of atoms	
counts	Counts	Dimensionless number of events.
counts/s	Counts per second	Number of events per second, equivalent to Hz. Customary equivalents "cps" or "c/s" *shall not* be used.
eV	electron volt	Energy, equivalent to $1.60217646 \times 10^{-19}$J.
%	Percent	Dimensionless ratio. Not to be used for concentrations (use mol%, vol%, wt%, etc.)
mol%	Molar/atomic percent	
vol%	Volumetric percent	
wt%	Weight percent	
mol_ppm	Molar parts per million	
vol_ppm	Volumetric parts per million	
wt_ppm	Weight parts per million	
mol_ppb	Molar parts per billion [*]	
vol_ppb	Volumetric parts per billion [*]	

wt_ppb	Weight parts per billion [*]	

** Parts per billion shall refer to short billions (10^9), never long billions (10^{12}).*

SI prefixes

Except where noted above, each unit supports the range of SI prefix codes, excluding centi (c), deci (d), deca (da) and hecto (h). These prefixes *should* only be used in cases where the prefix forms part of the widely accepted unit of measure for a particular quantity, such as the use of inverse centimetres (cm^{-1}) for measuring the wavenumber of light.

The supported range of prefix codes are:

Symbol	Magnitude	Note
Y	10^{24}	
Z	10^{21}	
E	10^{18}	
P	10^{15}	
T	10^{12}	
G	10^{9}	
M	10^{6}	
k	10^{3}	
m	10^{-3}	
u	10^{-6}	For compatibility reasons, the Latin character ‘u’ (U+0075) *should* be used in place of the Unicode micro sign ‘µ’ (U+00B5).
n	10^{-9}	
p	10^{-12}	
f	10^{-15}	
a	10^{-18}	

z	10^{-21}	
y	10^{-24}	

Appendix D - Unicode character substitutions

In the Unicode character set, there are several code points that produce visually indistinguishable glyphs. Consequently, to avoid confusion and maximise compatibility, the lowest code point *shall* be used in these cases. A non-exhaustive list of the required character substitutions are provided below:

- For the Ångström symbol, the Latin capital A with ring above ‘Å’ (U+00C5) *shall* be used, and not the Unicode Ångström symbol ‘Å’ (U+212B).
- For Kelvin, the Latin capital letter ‘K’ (U+004B) *shall* be used, and not the Unicode Kelvin symbol ‘K’ (U+212A).
- For ‘Ω’, the Greek capital letter omega ‘Ω’ (U+03A9) *shall* be used, and not the Unicode Ohm symbol ‘Ω’ (U+2126).
- For ‘µ’, the Unicode micro sign ‘µ’ (U+00B5) *shall* be used in place of the Unicode Greek small letter mu ‘µ’ (U+03BC).

Appendix E - Example files

SEM-XEDS hyperspectral map

This example represents a typical XEDS spectral map, as captured on an SEM. A *baseline* example of the same map, excluding all optional conditions and metadata, is provided thereafter.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<MSAHyperDimensionalDataFile Version="1.0" UID="7FE6B4B91EB3B81E"
xml:lang="en-US">
    <Header>
        <Title>Gneiss</Title>
        <Date>2012-08-15</Date>
        <Time>16:15:16</Time>
        <Timezone>UTC+10 AUS Eastern Standard Time</Timezone>
        <Author>
           Microbeam Laboratory Team; Acme Mineralogy Labs Inc.
        </Author>
        <Owner>Acme Mineralogy Labs Inc.</Owner>
        <Checksum Algorithm="SHA-1">
           79C5C30510A4F515E62F9F8BC9762BB8F59CF6ED
        </Checksum>
    </Header>
    <Conditions>
        <Instrument>
            <Manufacturer>Vandalay Industries</Manufacturer>
            <Model>Model 400 FEG-ESEM</Model>
        </Instrument>
        <Probe Class="EM">
            <BeamVoltage DataType="float" Unit="kV">15.</BeamVoltage>
        </Probe>
        <Acquisition Class="RegularArray/XY">
            <StepSize DataType="float" Unit="um">3.36</StepSize>
            <DwellTime DataType="float" Unit="ms">272.</DwellTime>
            <RasterMode>Probe</RasterMode>
        </Acquisition>
        <Detector Class="Spectrometer/XEDS">
            <Manufacturer>Acme Detector Co.</Manufacturer>
            <Model>Acme Model 1</Model>
            <MeasurementUnit>counts</MeasurementUnit>
```

```
            <Technology>SDD</Technology>
            <Calibration Class="Linear">
                <Quantity>Energy</Quantity>
                <Unit>eV</Unit>
                <Gain DataType="float">10.</Gain>
                <Offset DataType="float">-475.</Offset>
            </Calibration>
            <NominalThroughput DataType="float" Unit="kcounts/s">
               60.
            </NominalThroughput>
            <TimeConstant DataType="float" Unit="us">
               16.700001
            </TimeConstant>
            <StrobeRate DataType="float" Unit="Hz">1000.</StrobeRate>
            <Area DataType="float" Unit="mm2">10.</Area>
            <Elevation DataType="float" Unit="degrees">45.</Elevation>
        </Detector>
    </Conditions>
    <Data>
        <RegularArray Name="EDS map">
            <DataOffset DataType="int64">8</DataOffset>
            <DataLength DataType="int64">419225600</DataLength>
            <DatumType>byte</DatumType>
            <DatumDimensions>
                <Dimension DataType="int">2047</Dimension>
            </DatumDimensions>
            <CollectionDimensions>
                <Dimension DataType="int">512</Dimension>
                <Dimension DataType="int">400</Dimension>
            </CollectionDimensions>
        </RegularArray>
    </Data>
</MSAHyperDimensionalDataFile>
```

The same file, stripped of all conditions and header metadata, produces the following baseline file. Note that to use this file, the user must manually keep track of EDS gain & offset, beam current & voltage, etc.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<MSAHyperDimensionalDataFile Version="1.0" UID="1801E95BD3570275"
xml:lang="en-US">
    <Header />
    <Conditions />
    <Data>
        <RegularArray Name="EDS map">
            <DataOffset DataType="int64">8</DataOffset>
            <DataLength DataType="int64">419225600</DataLength>
            <DatumType>byte</DatumType>
            <DatumDimensions>
                <Dimension DataType="int">2047</Dimension>
            </DatumDimensions>
            <CollectionDimensions>
                <Dimension DataType="int">512</Dimension>
                <Dimension DataType="int">400</Dimension>
            </CollectionDimensions>
        </RegularArray>
    </Data>
</MSAHyperDimensionalDataFile>
```